НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

До захисту допущено:

В.о.завідувача кафедри

_____ Оксана ТИМОЩУК

«___»_____20__ р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системний аналіз і управління»

спеціальності 124 «Системний аналіз»

на тему: «Прогнозування часових рядів за допомогою рекурентних нейронних мереж LSTM»

Виконав:
студент IV курсу, групи КА-64
Безбах Володимир Павлович _____

Керівник:
доцент, к.ф-м.н. Яковлева А.П _____

Консультант з економічного розділу:
доцент, к.е.н. Шевчук О.А. _____

Консультант з нормоконтролю:
доцент, к.т.н. Коваленко А.Є. _____

Рецензент:
доцент, к.ф-м.н. Ільєнко А.Б. _____

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 124 «Системний аналіз»
Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ
В.о.завідувача кафедри
_____ Оксана ТИМОЩУК
«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту
Безбаху Володимиру Павловичу

1. Тема роботи «Прогнозування часових рядів за допомогою рекурентних нейронних мереж LSTM», керівник роботи Яковлева Алла Петрівна, к.ф-м.н., затверджені наказом по університету від «_25_»_травня_ 20_20_ р. №_1143-с__

2. Термін подання студентом роботи 08 травня 2020 року _____

3. Вихідні дані до роботи

1. Операційна система MacOs Catalina
2. Частота процесора 2.3 ГГц
3. Мова програмування Python
4. Середовище розробки – PyCharm Community Edition 2018.2.3
5. Бібліотеки, що використовувалися: NumPy, StatsModels, Pandas, Sklearn, Keras, Tensorflow, Matplotlib, Math

4. Зміст роботи

1. Проаналізувати існуючі математичні моделі
2. Проаналізувати моделі побудови нейронних мереж з довгою короткостроковою пам’яттю
3. Розробити програмний продукт для моделювання нейронних мереж з довгою короткостроковою пам’яттю
4. Розробити систему оцінки побудованої моделі
5. Виконати економічний аналіз програмного продукту

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)

1. Презентація

6. Консультанти розділів роботи*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	к.е.н., доц. Шевчук О. А.	21.04.20	28.05.20

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання	14.04.20	
2	Аналіз існуючих моделей	21.04.20	
3	Підбір показників для оцінки моделі	29.04.20	
4	Підбір вхідних даних для моделі	5.05.20	

* Якщо визначені консультанти. Консультантом не може бути зазначено керівника дипломної роботи.

5	Розробка продукту для моделювання та системи оцінювання моделі	15.05.20	
6	Тестування продукту, отримання результатів	18.05.20	
7	Оформлення дипломної роботи	20.05.20	

Студент Володимир Павлович БЕЗБАХ

Керівник Алла Петрівна ЯКОВЛЕВА

РЕФЕРАТ

Дипломна робота: 94 с., 34 рис., 6 табл., 2 дод., 15 джерел.

НЕЙРОННІ МЕРЕЖІ, НЕЙРОННІ МЕРЕЖІ З ДОВГОЮ КОРОТКОСТРОКОВОЮ ПАМ’ЯТТЮ, ОЦІНКА, ЧАСОВІ РЯДИ

До бакалаврської дипломної роботи Безбаха Володимира Павловича на тему: «Прогнозування часових рядів за допомогою рекурентних нейронних мереж LSTM»

Дана робота присвячена аналізу і дослідженню роботи нейронних мереж з довгою короткостроковою пам’яттю на прикладі задачі прогнозування часових рядів.

Метою роботи є побудова моделі нейронної мережі з довгою короткостроковою пам’яттю для прогнозування часових рядів, оцінка роботи моделі за допомогою статистичних показників.

Об’єктом дослідження є часові ряди, модулі нейронних мереж, нейронні мережі з довгою короткостроковою пам’яттю.

ABSTRACT

Bachelor thesis: 94 p., 34 fig., 6 tabl., 2 add. and 15 references

EVALUATION, NEURAL NETWORKS, NEURAL NETWORKS WITH LONG SHORT-TERM MEMORY, TIME SERIES

To the bachelor's thesis of Bezbakh Volodymyr on the topic: "Forecasting time series using recurrent neural networks LSTM"

This paper is devoted to the analysis and study of neural networks with long short-term memory on the example of the problem of time series prediction.

The aim of the work is to build a model of a neural network with long short-term memory for forecasting time series, evaluation of the model using statistical indicators.

The object of research is time series, modules of neural networks, neural networks with long short-term memory.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

АР – процес авторегресії

КС – процес ковзного середнього

МНК – метод найменших квадратів

DW – значення критерію Дарбіна-Уотсона для моделі

LSTM – нейронні мережі з довгою короткостроковою пам’яттю

MAE – значення середня абсолютної помилки для моделі

MAPE – значення середня абсолютної помилки у відсотках для моделі

RMSE – квадратний корінь середньої квадратичної різниці між прогнозованими значеннями та фактичними спостереженнями для моделі

RNN – рекурентні нейронні мережі

ВСТУП

Задачі прогнозування та передбачення постають перед нами кожен день. Математичне моделювання, що дозволяє зробити прогноз на майбутнє, оцінити його якість та користуватися отриманим прогнозом завжди буде актуальною та потрібною задачею. У наші часи розвиток потужності для обрахування росте невпинно, відкриваючи все нові і нові можливості для моделювання, а також прогнозування та аналізу різноманітних процесів, в якості яких у даній роботі розглядаються часові ряди.

Традиційні методи математичного моделювання, такі як статистичні моделі, що будуть розглянуті у роботі, часто вимагають вироблених вручну особливостей та експертних знань у цій галузі. Вони часто вимагають або строгої, або слабкої стаціонарності даних, що практично неможливо отримати на практиці. Це означає, що перш ніж застосовувати класичні статистичні моделі, їх потрібно перетворити та подати у належному вигляді, але самі методи перетворення даних представляють власний клас задач, який необхідно розв’язати.

Нейронні мережі такі проблеми оминають, оскільки вони нелінійні за своєю суттю. Хоча нейронні мережі існують з 1950-х і 1960-х років, лише останнім часом вартість збору та аналізу даних стала дешевшою, ніж раніше через появу хмарних обчислень, а також дешевшого і більшого обладнання для зберігання даних. Масовий збір даних, що випливав з цього, викликав відновлення інтересу до алгоритмів, керованих даними, з яких нейронні мережі є прикладом. Нейронні мережі вчяться моделювати процеси на основі прикладних вхідних та вихідних значень, а продуктивність покращується, коли в навчанні моделей використовується більше даних. Це зміщує необхідні експертні знання, необхідні для класичних статистичних моделей, з вміння пристосовувати дані на знання алгоритмів нейронних мереж. Саме тому метою

роботи є побудова моделі нейронної мережі з довгою короткостроковою пам’яттю для прогнозування часових рядів, оцінка роботи моделі за допомогою статистичних показників. Об’єктом дослідження виступають часові ряди, модулі нейронних мереж, нейронні мережі з довгою короткостроковою пам’яттю. Постановка задачі для даної роботи – аналіз і дослідження роботи нейронних мереж з довгою короткостроковою пам’яттю на прикладі задачі прогнозування часових рядів, розробка системи оцінки побудованої моделі.

Таким чином у роботі розглядаються класичні статистичні методи моделювання, такі як процеси з ковзним середнім (КС), авторегресивні моделі (АР) та інші моделі, а також дається огляд роботи нейронних мереж, рекурентних нейронних мереж та їх особливого виду – нейронні мережі з довгою короткостроковою пам’яттю, які дозволяють вирішити задачі, з якими не в змозі впоратися звичайні рекурентні нейронні мережі через втрату можливості встановлення зв’язків між подіями, які відбувалися через деякий проміжок часу.

1 ОСОБЛИВОСТІ РОЗВИТКУ ПРОГНОЗУВАННЯ ЧАСОВИХ РЯДІВ

1.1 Особливості прогнозування часових рядів

Прогнозування є одною з найскладніших і найпотрібніших задач аналізу даних. Складність процесу прогнозування пов’язана з необхідністю аналізу і оцінювання великих обсягів даних, ускладненням методів, появою концептуально нових підходів до прогнозування процесів різної природи тощо. Тому на сьогодні стан розвитку методів прогнозування тісно пов'язаний з розвитком інформаційних технологій. Так звані, інформаційні системи прогнозування, що відображають цей зв'язок в рамках економетрики, фінансової математики, статистики, набувають свого прояву в широкому спектрі прикладних галузей науки, а також у сферах виробництва, фінансового планування в економіці і торгівлі. Сьогодні вони є невід’ємними складовими процесів управління складними системами і прийняття управлінських рішень, застосовуються аналітиками для оцінювання ризиків фінансового інвестування тощо. [1]. Прогнозування часових рядів є актуальною науковою проблемою, що має безліч застосувань у теорії управління, в економіці, медицині, фізиці та інших галузях. Якщо казати про нейромережеві методи, то вони добре себе зарекомендували як засіб моделювання динамічних систем при невідомій апріорі математичній моделі динамічної системи. Нейронна мережа може бути навчена на відомих прикладах реалізацій динамічного процесу і потім використовуватися для прогнозування на нових даних. [2]. У класичній статистичній обробці даних часових рядів прогнозування майбутнього називається екстраполяцією. У наші часи, сучасні галузі звертають на цю тему (тему прогнозування часових рядів) все більше та більше уваги. Сенс прогнозування полягає у побудові і розробці моделей, що відповідатимуть спостереженим даним та подальшому використанні цих моделей для отримання

майбутніх значень. Особливість прогнозування – прийняття факту, що дані у майбутньому доступними бути не можуть і їх можливо оцінити тільки завдяки тим даним, що вже є у наявності, тобто даним, що збираються з подій, що вже відбулися[3].

Прогнозування за допомогою часових рядів – один з найпопулярніших підходів до прогнозування розвитку різноманітних процесів, об'ємів торгових операцій, об'ємів виробництва та накопичення продукції на складах, вартості фінансових активів, розподілу світової маси грошей, оцінювання альтернативних економічних стратегій та ін. [4]. Наприклад, у фінансовій сфері часто розглядається проблема прогнозування короткочасної тенденції часового ряду без розрахунку безпосередньо прогнозних значень. Це пов'язано з тим, що фінансовим часовим рядам часто притаманний нестійкий характер коливань. Такі ряди можуть бути близькими до випадкових. В цьому випадку будують моделі, які б дозволяли розрахувати прогнози знаків приростів значень часового ряду на одну точку вперед з необхідною максимальною точністю. Особливістю таких моделей і методів є можливість знаходження достатньо стійкої залежності поточного спостереження від попередніх. [5].

Особливість прогнозування часових рядів полягає в тому, що аналізуються лише дані спостережень без додаткової інформації, без аналізу впливу зовнішніх факторів. Такий аналіз виглядає досить неповним, але доволі часто прогнози часових рядів є точними. Для розробки системи прогнозування часового ряду можливе використання безлічі алгоритмів і методів, які відрізняються складністю реалізації, рівнем достовірності, кількістю обчислень, величиною часових втрат та ін. Задача прогнозування значень, тренду та характеристик часового ряду є надзвичайно актуальною через те, що саме часові ряди можуть бути і успішно використовуються для моделювання процесів, приклади яких зазначено вище.

1.2 Методи побудови математичних моделей для моделювання

До типів моделювання, що використовуються для опису часових рядів, можна віднести методи інтелектуального аналізу, наприклад алгоритми нечіткого виведення на основі бази правил, дерева рішень, генетичні алгоритми, еволюційне програмування, метод групового урахування аргументів, штучні нейронні мережі, баєсівські мережі і ін. Найбільш поширені методи моделювання часових рядів є класичні регресивні моделі, авторегресивні моделі з урахуванням ковзного середнього і їх різноманітні доповнення і модифікації. Для багатьох реальних динамічних процесів ці моделі можуть не давати надточні результати через складну поведінку процесів, що розглядаються. В такому випадку обирають інші моделі, які можуть врахувати складну поведінку процесів.

Для прогнозування і моделювання процесів, з якими неточно працюють регресійні моделі, можуть використовуватись ймовірнісні методи. Одним із прикладів таких моделей можуть бути мережі Байєса, що є інструментом інтелектуального аналізу даних, до задач якого можна віднести прогнозування часових рядів. Для цілей прогнозування можна використовувати Метод Групового Урахування Аргументів (МГУА), що заснований на рекурсивному селективному відборі моделей, на основі яких будуються складніші моделі. Точність моделювання на кожному наступному кроці рекурсії збільшується за рахунок ускладнення моделі. Традиційні статистичні моделі не завжди підходять для описання реальних процесів, таких як динаміка об’ємів виробництва та накопичення продукції на складах, вартості фінансових активів та ін., так як такі процеси у загальному випадку є істотно нелінійними і часто мають основу, близьку до хаотичної, але віддалено може нагадувати квазіперіодичну або змішану, за значеннями деяких параметрів [6]. В такому

випадку вибір штучних нейронних мереж для аналізу і прогнозування буде одним з найбільш вдалих. [7-8].

Нейронна мережа – це сукупність нейронних елементів і зв'язків між ними. Основний елемент нейронної мережі – це формальний нейрон (Рисунок 2), що здійснює операцію нелінійного перетворення суми добутків вхідних сигналів на вагові коефіцієнти. Найпростіша форма штучної нейронної мережі - це одношаровий перцептрон. П.Розенблат розробив такі види перцептронів у 1950х роках [8]. Математичне трактування одношарового перцептрона може бути описае наступною формулою

$$f(\bar{x}, \bar{w}, b) = \begin{cases} 0, \text{якщо } \langle \bar{w}, \bar{x} \rangle + b \leq 0 \\ 1, \text{якщо } \langle \bar{w}, \bar{x} \rangle + b > 0 \end{cases}, \quad (1.1)$$

де $\bar{x}$ – вектор вхідних сигналів;

$\bar{w}$ – вектор вектор вагів;

b – зміщення (поріг збудження) персептрона.

При переході до неперервної функції буде можливо використовувати похідну для навчання вектора вагів $\bar{w}$ та зміщення b. Початково неперервний перцептрон був спроектований з використанням сигмоїди (Рисунок 1.1):

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (1.2)$$

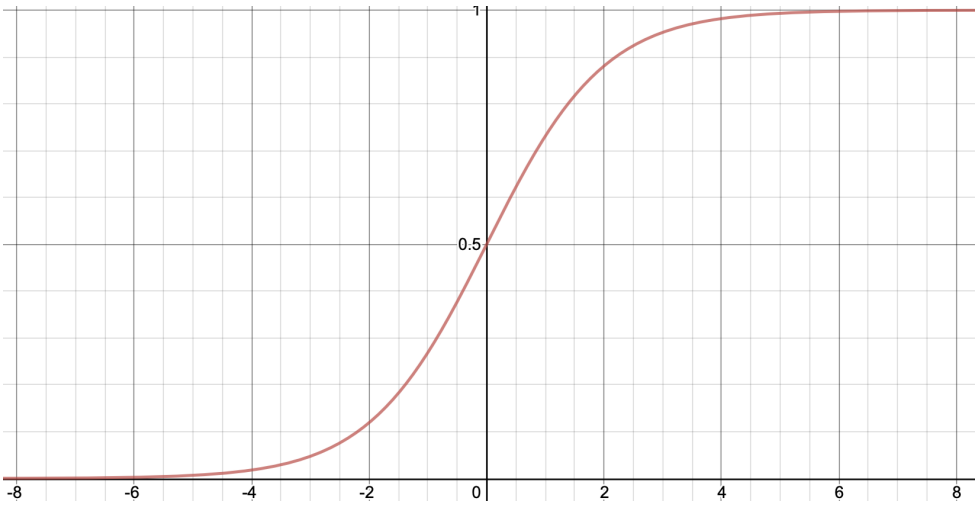


Рисунок 1.1 – Графік активаційної функції $\sigma(x)$

Сигмоїда в даному випадку називається функцією активації або активаційною функцією. Результуючий сигнал перцептрона, що використовує сигмоїду, як функцію активації, може бути описаний наступною формулою

$$f(\bar{x}, \bar{w}, b) = \sigma(\langle \bar{w}, \bar{x} \rangle + b) = \frac{1}{1 + e^{-\langle \bar{w}, \bar{x} \rangle - b}}, \tag{1.3}$$

де $\bar{x}$ – вектор вхідних сигналів;

$\bar{w}$ – вектор вектор вагів;

b – зміщення (поріг збудження) персептрона.

Продемонструємо графічне представлення математичної моделі одношарового перцептрона (Рисунок 1.2):

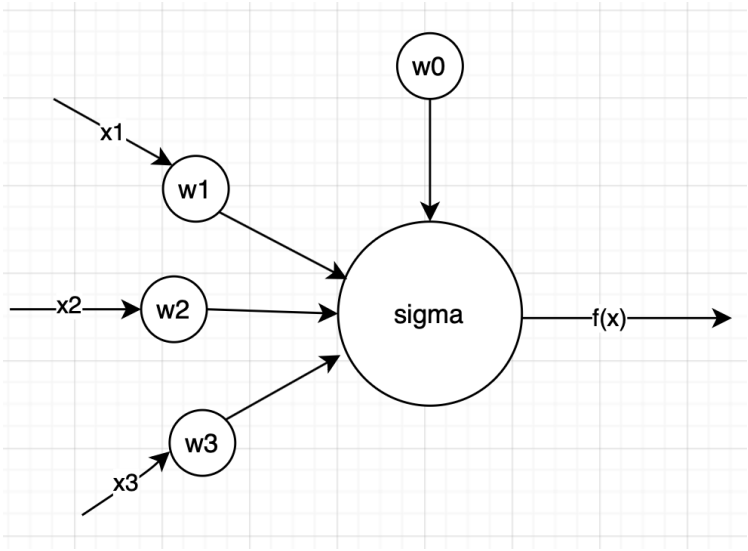


Рисунок 1.2 – Математична модель одношарового перцептрона

Одношаровий перцептрон може бути покращений до багатошарового випадку який називають багатошаровий перцептрон (Рисунок 1.3). У цьому випадку мвємо кілька шарів, кожен з яких складається з декількох нейронів. Інформація тече лише в одному напрямку в багатошаровому перцептроні, і, таким чином, багатошаровому перцептрони ациклічні. У випадку багатошарового перцептрона у нас є вхідний шар, вектор $\bar{x}$, середні шари і, нарешті, вихідний шар. Шари посередині називаються прихованими шарами, оскільки вони не є ні частиною вводу, ні виводу. Кожен прихований шар повністю пов'язаний з попереднім шаром і вихідний шар повністю пов'язаний з останнім прихованим шаром. Кожен прихований шар і вихідний шар має фіксовану кількість нейронів, яка може бути налаштована параметрами моделі. Мета цієї мережі – апроксимувати функцію f.

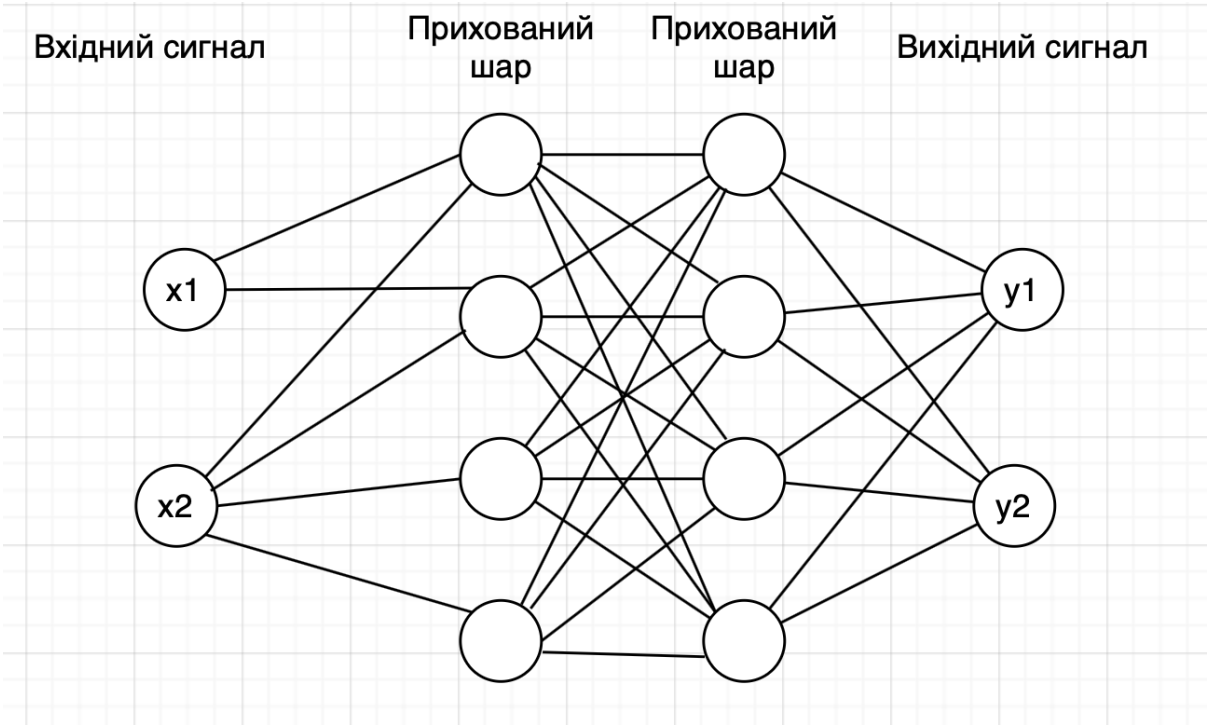


Рисунок 1.3 – Приклад нейронної мережі – багатошаровий персептрон

Методи штучних нейронних мереж довели раціональність їх використання своєю високою ефективністю при дослідженні різноманітних систем. Штучні нейронні мережі відносяться до області neural computing, що є областю обчислювальних технологій, яка швидко розвивається. Операції обчислення в цих мережах виконуються за допомогою великої кількості елементарних частин для обчислення. Структура системи нагадує мережу, що задається графом, у якому обчислювання виконуються в певних вузлах, а от інформація передається за напрямком дуг. Кожен елемент (тобто вузол) графа є нейроном, елементом обчислення, який у сукупності з іншими елементами утворює нейронну мережу. Аналогом такого вузла у фізіологічній нервовій системі є нервова клітина мозку [9].

У загальному випадку штучна нейронна мережа є адаптивною нелінійною динамічною системою. За допомогою рівноважних станів такої мережі можна вирішувати математичні або обчислювальні задачі, такі як апроксимація функцій, класифікація, кластеризація, прогнозування та багато інших. При

вирішенні завдань з використанням нейронних мереж підбирають стандартну конфігурацію нейронної мережі, але з урахуванням складності і особливості завдання підбір існуючих конфігурацій може бути проблематичний. Якщо ж завдання не може бути зведене до жодного з відомих типів нейронні мережі, доводиться вирішувати складну проблему синтезу нової конфігурації. Для визначення структури моделі нейронної мережі необхідно вирішити кілька завдань: побудувати класифікацію нейронних мереж; провести аналіз існуючих нейронних мереж; розробити основні критерії відбору нейронних мереж для побудови моделі; визначити основні характеристики для визначення якості моделі на основі нейронної мережі. Основною характеристикою нейронної мережі є модель мережі.

Охарактеризувати нейронні мережі можна за видами нейронів , що використовуються в мережі, структури моделі мережі, способам навчання мережі, завданням які вирішує мережа. Розглядаючи завдання, що вирішуються нейронними мережами можна виділити широке коло завдань обробки і аналізу даних - розпізнавання і класифікація образів, прогнозування, управління, кластерний аналіз, апроксимація, нейромережеве стиснення даних, асоціативну пам'ять і т.д. [10]. Нейронні мережі можна поділити, наприклад, за видами навчання – мережі, що навчаються з учителем та мережі, що навчаються без учителя. Нейронні мережі, що навчаються з учителем використовуються для отримання даних про внутрішні взаємодії у мережах. Внутрішні взаємодії описуються математичними рівняннями, визначення яких дозволяє потім їх ефективно використовувати у задачах прогнозу та передбачення. В даному випадку у ролі учителя використовується набір параметрів, що розміщується на виході мережі. На вхід мережі при цьому подається відповідний даному виходу вхідний набір даних. Мережа навчається встановлювати залежність між початковою інформацією і результатами її навчання [11]. Нейронна мережа, що навчається без учителя, працює на основі навчання змаганням. При такому навчанні відбувається наступне: на вхід нейронної мережі подається набір даних

і вузли, зазначені вище, «змагаються» один з одним за якнайкращу відповідність вихідному набору згідно вибраним критеріям.

В результаті змагання визначається переможець, який дає змогу коригувати структуру нейронної мережі. Клас нейронних мереж, що самоорганізуються, навчаються без вчителя, позначається терміном адаптивні нейронні мережі. Важливою особливістю методу адаптивних нейронних мереж є те, що він є чисельним, а не символьним методом обробки даних. Однією з унікальних особливостей адаптивних нейронних мереж є те, що вони надають внутрішньо властиві точні і прості механізми для поділу обчислювального завдання на окремі потоки, що дозволяє проводити обчислення з високим ступенем паралельності.

1.3 Відмінність нейронних мереж з довгою короткостроковою пам’яттю від звичайних мереж

Звичайні нейронні мережі не мають можливості аналізувати події, які відбувалися в минулому. Цю проблему намагаються вирішити за допомогою рекурентних нейронних мереж (RNN). Даний тип нейронних мереж має особливість – наявність зворотних зв’язків (на відміну від тих, що вже було розглянуто), що дозволяє їм зберігати інформацію. Рекурентна нейронна мережа може бути представлена як послідовність одношарових елементів, які передають результуючий сигнал наступній копії. Рекурентні нейронні мережі можуть бути використані для аналізу і прогнозування часових рядів, але вони мають суттєвий недолік – при збільшенні відстані між суттєвими для прогнозування шарами (тобто при збільшенні часу виявлення залежності), ці нейронні мережі втрачають можливість зв’язувати інформацію.

Існує модифікація рекурентних нейронних мереж, що не має проблем звичайних рекурентних нейронних мереж – у даній модифікації нейрони мають змогу запам’ятовувати інформацію, що була отримана нещодавно, але не мають можливості надовго зберегти в пам'яті щось, що обробили багато циклів назад, якою б важливою та інформація не була. У LSTM-мережах внутрішні нейрони представляють собою складну систему, що будується складним чином, який буде розглянуто пізніше. Основна ідея цих мереж – передача пам’яті між шарами, для цієї цілі в склад кожного нейрона включається спеціальний показник, який називається «стан клітини». Він проходить через всю мережу, збережуючи інформацію у собі і передаючи її на наступні ітерації виконання та навчання. Особливими елементами нейрона такої мережі також є фільтри, які визначають, як поводитись з інформацією, що проходить через нейрон. Фільтри приймають рішення щодо видалення, зберігання чи подачі на вихід інформації.

Кожний нейрон має бути обладнаний «станом», що і грає роль пам’яті у цій мережі, а також основними типами фільтрів - вхідним, вихідним і забуваючим (останній ще можна визначати як фільтр забувуання). Метою цих фільтрів є захист інформації. Вхідний фільтр визначає, скільки інформації з попереднього шару буде зберігатися в клітці. Вихідний фільтр визначає, скільки інформації отримають наступні шари. Якщо казати про фільр забування, то він визначає, яку інформацію потрібно не опрацьовувати при переході до наступного етапу алгоритма.

1.4 Огляд існуючих програмних засобів прогнозування часових рядів

Azure Time Series Insights – це повністю керована служба аналітики, зберігання та візуалізації для керування даними часових рядів масштабу інтернету речей (IoT) у хмарному сховищі. Забезпечує масштабування даних

часових рядів та дає змогу за лічені секунди досліджувати та аналізувати мільярди подій, що передаються з усього світу.

Eviews – це сучасний економетричний, статистичний та прогнозний пакет, який пропонує потужні аналітичні інструменти в рамках гнучкого, простого у користуванні інтерфейсу. Дозволяє швидко та ефективно керувати своїми даними, виконувати економетричний та статистичний аналіз, генерувати прогнози чи моделювання моделей та створювати високоякісні графіки та таблиці для публікації чи включення в інші додатки.

Anodot – використовує алгоритми машинного навчання для постійного аналізу та співвіднесення кожної бізнес метрики, надаючи сповіщення та прогнози в реальному часі.

STATISTICA – надає потужні і зручні у використанні інструменти для статистичного і графічного аналізу, прогнозування, data mining, створення власних користувальницьких додатків, інтеграції, спільної роботи, web-доступу та ін.

Абсолютно всі перераховані програмні продукти є комерційними і доступні після оплати у повному обсязі, або ж надають безкоштовний функціонал, який є значно обмеженим у порівнянні з платними версіями. Такі програмні продукти не можуть бути використані в особистих цілях безкоштовно, що суттєво ускладнює використання сучасних технологій прогнозування і аналізу часових рядів за допомогою штучного інтелекту.

1.5 Висновки до розділу

Проаналізувавши потреби сучасного інформаційного аналізу бачимо, що не дивлячись на величезне різноманіття програмного забезпечення та різноманітних програмних продуктів для аналізу та передбачення часових рядів

ця тема є актуальною і нині через постійну мінливість задач прогнозування, процесу технічного розвитку суспільства, розповсюдження прикладного використання досліджуваних процесів (тобто часових рядів) на об’єкти досліджень, що раніше не аналізувалися за допомогою математичних моделей, але було отримано таку можливість через надзвичайний зріст обчислювальних потужностей та технологічний прогрес, що невпинно веде до подальшого розвитку обраної тематики прогнозування та аналізу.

Аналіз існуючого програмного забезпечення показав наявність спеціалізованих програмних продуктів, що вирішують поставлену задачу прогнозування часових рядів, мають складну архітектуру та комерційний успіх – але останній пункт свідчить і про те, що це програмне забезпечення не є безкоштовним, що не дає змогу використовувати всі його можливості у навчальних цілях у повному обсязі. Розглянувши особливості побудови математичних моделей для аналізу та прогнозування часових рядів, можемо зробити висновок, що для досягнення поставленої цілі може бути використано декілька способів математичної побудови моделей, апроксимації та прогнозування. Пропонується варіант реалізації поставленої задачі прогнозування часового ряду за допомогою рекурентної нейронної мережі з довгою короткостроковою пам’яттю (LSTM). Обраний варіант реалізації вирішення проблеми пропонує лише один метод вирішення задачі аналізу, який може бути використаний у подальших дослідженнях і експериментах як і дослідниками – аналітиками, так і звичайними користувачами.

2 ОСНОВНІ СТАТИСТИЧНІ МОДЕЛІ, НЕЙРОННІ МЕРЕЖІ З ДОВГОЮ КОРОТКОСТРОКОВОЮ ПАМ'ЯТТЮ

2.1 Основні задачі аналізу та прогнозування часових рядів

Задача аналізу часового ряду виникає в ситуаціях, які потрібно відрізняти від простіших задач аналізу. Прикладом таких задач аналізу може бути задача, в якій потрібно аналізувати дані та спостереження при відсутності їх природніх надходжень.

Основні відмінності часових рядів від простих стохастичних вибірок полягають у:

- пов'язанності послідовних у часі показників часових рядів;
- інформаційна значущість показників типового часового ряду спадає при достатньому віддаленні цільового значення від шуканого (або спостережуваного) значення;
- збільшення кількості показників часового ряду не впливає на точність статистичних характеристик;
- при наявності закономірностей у часових рядах точність визначення статистичних характеристик зменшується;

В основі процесу побудови часових рядів є можливість і потреба співставлення його окремих показників. Ці фактори досягаються при умові того: що всі елементи ряду характеризують явище, характеристики якого можливо представити у дискретному наборі даних на часових проміжках. При цьому, кожне значення кожного показника такого явища необхідно фіксувати через однакові часові інтервали, а час спостереження неодмінно має бути достатньо великим для виявлення стійкої тенденції окремого процесу.

Виділяють одномірні часові ряди, отримані при фіксованій кількісній характеристиці, і багатовимірні часові ряди, отримані при спостереженні декількох характеристик виділеного об'єкта. Часові ряди можуть бути

дискретними і неперервними. У складі дискретних рядів виділяють ряди для рівновіддалених моментів спостереження і для довільних моментів спостереження. Часові ряди також бувають детермінованими і випадковими: перші отримують на основі значень деякої невипадковою функції; другі - є результат реалізації деякої випадкової величини. Іншою характеристикою часового ряду є гетероскедастичність, що означає, що дисперсія часового ряду є функцією від часу.

Тренди часового ряду також можуть бути різними - лінійними або нелінійними, детермінованими або стохастичними. Гетероскадестичні процеси з трендами характерні для нестійкої шводкоростущої перехідної економіки. Прогноз часового ряду або процесу може бути представлений точним значенням змінної, інтервалом, в який ця змінна потрапить або ж ймовірністю набування змінної певного значення у певному інтервалі. Задачу прогнозування в залежності від поставленої задачі управління та мети прийняття рішення можна ставити по-різному.

Прогнозування може стосуватись наступних складових процесу [12]:

- прогнозування детермінованого тренду, як індикатора довгострокових змін процесу;
- випадкового (нерегулярного) тренду, як показника коротко- та середньострокових змін;
- короткострокових змін, тобто, прогнозування коливань (відхилень), що накладаються на тренд;
- сезонних ефектів;
- приростів (швидкості) зміни процесу, які визначаються першими різницями;
- дисперсії або стандартного відхилення, як міри розсіювання процесу (наприклад, волатильність, яку часто використовують за міру ризику у інвестуванні або міру якості на виробництві).

Задача побудови будь-якої моделі (математичної, логічної або ймовірнісної) ставиться в залежності від того, які складові процесу необхідно прогнозувати. Побудова моделі має на меті отримати високу якість прогнозу за заданий період часу.

Для прикладу розглянемо два типу тренда: детермінований та стохастичний. Детермінований тренд можна представити у вигляді поліноміальної залежності від часу деякого порядку:

$$y(k) = a_0 + a_1 \cdot k + a_2 \cdot k^2 + a_3 * k^3 + \cdots + a_l \cdot k^l + \varepsilon_k$$
$$E[\varepsilon_k] = 0, \quad (2.1)$$

де $a_0, \dots, a_l -$ коефіцієнти ряду;

$k, \dots, k^l -$ дискретний час;

$\varepsilon_k -$ випадковий шум;

$E -$математичне сподівання.

У цьому випадку прогноз зводиться до підстановки у формулу часу k і знаходження характеристик та значення тренду у цей момент. Стохастичний тренд можна представити у вигляді рівняння випадкового кроку зі зміщенням:

$$y(k) = a_0 + a_1 \cdot y(k-1) + \varepsilon_k$$
$$E[\varepsilon_k] = 0 \quad (2.2)$$

2.2 Деякі статистичні методи прогнозування часових рядів

2.2.1 Метод найменших квадратів

Один з простих методів прогнозування часових рядів - це застосування алгоритму МНК. Ми припускаємо, що наш часовий ряд можна представити у наступному матричному вигляді:

$$Y = X \cdot \theta + \varepsilon, \tag{2.3}$$

де Y – вектор спостережень;

X – матриця вимірів;

θ – вектор параметрів (коефіцієнтів) моделі;

ε - вектор похибок.

Для використання цієї моделі ми маємо пристосувати її до наших даних, з яких ми будемо отримувати неупереджені оцінки $\hat{\theta}$ параметрів θ. У нашому наближенні ми хочемо мінімізувати суму залишків у квадраті

$$S(\theta) = \|Y - X \cdot \theta\|_2^2$$

Це досягається при наступному вигляді вектора неупереджених оцінок:

$$\hat{\theta} = (X^T X)^{-1} X^T Y$$

Для прикладу розглянемо наступний варіант задання моделі:

$$Y_t = \theta_1 + X_1 \cdot \theta_2 + \varepsilon_t$$

Або у матричному вигляді

$$Y = \begin{pmatrix} 1 & x_1 \\ \dots & \dots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} \theta_1 \\ \theta_2 \end{pmatrix} + \varepsilon \qquad (2.4)$$

Після приведення цієї моделі до вигляду, в якому можна використовувати формулу для пошуку вектора неупереджених оцінок, ми отримаємо вектори параметрів θ_1 та θ_2. Метод МНК працює дуже добре для часових рядів з лінійним трендом, але не зможе визначити сезонність часового ряду, оскільки вся ця інформація буде прихована у векторі ε.

2.2.2 Поліноміальна регресія

Ми можемо модифікувати метод МНК, щоб він відповідав поліноміальним моделям. Знову припускаємо, що наш часовий ряд ряд можна представити у наступному матричному вигляді:

$$Y = X \cdot \theta + \varepsilon$$

У випадку поліноміальної регресії матриця X буде мати наступний вигляд:

$$X = \begin{pmatrix} 1 & \cdots & x_1^m \\ \dots & \ddots & \vdots \\ 1 & \cdots & x_n^m \end{pmatrix}$$

Метод наближення, що використовувався для моделі МНК може бути використаний і в цьому випадку. Щоб правильно здійснити регресію за допомогою цієї моделі, потрібно заздалегідь знати розмірність полінома m.

2.2.3 Загальні лінійні процеси

Будь-який слабостаціонарний процес (тобто процес з постійним математичним очікуванням і дисперсією, в якому коваріація між значеннями ряду залежить тільки від величини часового здвигу між елементами) з нульовим математичним сподіванням можна представити у наступному вигляді: [13]

$$y(t) = \varepsilon_t + \sum_{i=1}^{\infty} w_i \varepsilon_{t-i}\,, \tag{2.5}$$

де ε_t – дискретний білий шум;

$w = (w_1, w_2, \dots)$ – вектор вагових коефіцієнтів.

Кількість ваг, необхідних для цієї моделі, нескінченна, і тому моделі в цій формі не є практичними. Припускаючи що лише кінцева кількість ваг не дорівнює нулю, ми можемо отримати нові моделі.

2.2.4 Процес ковзного середнього

У процесі ковзного середнього (КС) використовується зважена сума попередніх q дискретних точок даних білого шуму, яка записується як КС(q). Це вид загального лінійного процесу, у якому перші q елементів вектора w є ненульовими. Загальний вигляд моделі:

$$y(t) = c + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}\,, \tag{2.6}$$

де ε_t – дискретний білий шум;

$\theta_1, \dots, \theta_q$ – вагові коефіцієнти;

c – константа.

Щоб пристосувати цю модель до часовому ряду, нам потрібно знайти значення $c, \theta_1, \dots, \theta_q$ та σ^2 дискретного білого шуму ε_t.

2.2.5 Авторегресивна модель

У моделі АР кожен елемент часового ряду є лінійною комбінацією попередніх елементів, на відміну від зваженої суми дискретних точок даних щодо білого шуму, які використовує КС. Авторегресивна модель порядку p позначається АР(p) і має наступний загальний вигляд:

$$y(t) = c + \varepsilon_t + \sum_{j=1}^{p} \emptyset_j y(t-j)\,, \qquad (2.7)$$

де $\emptyset_1, \dots, \emptyset_p$ – параметри моделі;

ε_t – дискретний білий шум, ;

c – константа.

Знову ж таки, для пристосування цієї моделі до часовому ряду, нам потрібно знайти значення $c,\ \emptyset_1, \dots, \emptyset_p$ та σ^2 дискретного білого шуму ε_t.

2.2.6 Процес авторегресії – ковзного середнього

Моделі АР та КС можна вдосконалити, комбінуючи їх. Цей новий процес називається АР, АРКС (p, q) і має наступний загальний вигляд:

$$y(t) = c + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} + \sum_{j=1}^{p} \emptyset_j y(t-j)\,, \qquad (2.8)$$

де ε_t – дискретний білий шум;

$\theta_1, \dots, \theta_q$ – вагові коефіцієнти;

$\emptyset_1, \dots, \emptyset_p$ – параметри моделі;

c – константа.

Модель АРКС працює зі стаціонарними часовими рядами, але є модель, яка працює також і з нестаціонарними часовими рядами, це модель авторегресії з інтегрованим ковзним середнім – АРІКС, або АРІКС(p, q)

2.3 Деякі метрики для оцінки точності побудованих моделей

2.3.1 MAE

Середня абсолютна помилка (MAE) – вимірює середню величину помилок у наборі прогнозів, не враховуючи їх знак (тобто для даної метриуки немає різниці, модель схибила в додатну чи від’ємну сторону. Це середнє значення абсолютних різниць між прогнозованими значеннями та фактичними спостереженнями для тестової вибірки.

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|\,, \tag{2.9}$$

де y_i – фактичне значення у і – й момент часу;

$\hat{y}_i$ – значення прогнозованої моделі у і – й момент часу.

2.3.2 MAPE

MAPE – Середня абсолютна помилка у відсотках. Зазвичай використовується для оцінки точності прогнозу. Також ця метрика оцінює величину помилок порівняно зі значеннями ряду. Ця метрика використовується для порівняння різних моделей для одного ряду та оцінки економічного ефекту, за рахунок підвищення точності прогнозу.

$$\mathrm{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i}\cdot 100\%\,, \tag{2.10}$$

де y_i – фактичне значення у і – й момент часу;

$\hat{y}_i$ – значення прогнозованої моделі у і – й момент часу.

2.3.3 RMSE

RMSE – метрика, що вимірює квадратний корінь середньої квадратичної різниці між прогнозованими значеннями та фактичними спостереженнями для тестової вибірки.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}\,, \tag{2.11}$$

де y_i – фактичне значення у і – й момент часу;

$\hat{y}_i$ – значення прогнозованої моделі у і – й момент часу.

2.3.4 DW

DW – критерій Дарбіна-Уотсона для тестування автокореляції у залишках моделі.

$$\mathrm{DW} = \frac{\sum_{i=1}^{n}((\hat{y}_i - y_i) - (\hat{y}_{i-1} - y_{i-1}))^2}{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}, \tag{2.12}$$

де y_i – фактичне значення у і – й момент часу;

$\hat{y}_i$ – значення прогнозованої моделі у і – й момент часу.

Критерій приймає значення від 0 до 4, описуючи додатну кореляцію при $\mathrm{DW} = 0$ та від'ємну при $\mathrm{DW} = 4$. Для моделей з відсутністю автокореляції похибок $\mathrm{DW} = 2$.

2.4 Використання рекурентних нейронних мереж з довгою короткостроковою пам’яттю

2.4.1 Архітектура шарів у багатошаровому перцептроні

Традиційно всі нейронні мережі побудовані за допомогою нейронів, розділеними за шарами. Такий спосіб побудови дозволяє поширювати інформацію між шарами тільки у прямому напрямку. Розглянемо процес обрахунку вихідного сигналу за вектором вхідних сигналів більш детально.

Розглянемо багатошаровий перцептрон. У такому елементі нейронної мережі кожен найпростіший нейрон (тобто одношаровий перцептрон) має зв’язок у одному напрямку з кожним одношаровим перцептроном наступного шару. Кожен такий зв’язок має свою вагу. В такому випадку результуючий сигнал багатошарового перцептрона можна записати наступним чином:

$$x_{k+1} = g_k(W_k x_k + b_k)$$
$$y = w(x_{p-1}),$$

де $x_0 -$ вхідний вектор;

$y -$ вихідний вектор (результуючий сигнал) ;

$W_k -$ матриця вагів k-го шару;

$(p - 1)$ – індекс останнього прихованого шару;

$b_k -$ вектор зміщення k-го шару;

$g_k -$ активаційна функція k-го шару;

$w -$ результуюча функція.

2.4.2 Активаційні функції

Як вже було сказано у першому розділі, в ролі початкової активаційної функції виступала сигмоїда, але зараз на практиці застосовуються різноманітні функції, окрім сигмоїди. Наприклад, розглядають як активаційну функцію тангенс гіперболічний (Рисунок 2.1):

$$g(x) = \tanh(x) = 2\sigma(2x) - 1 \tag{2.13}$$

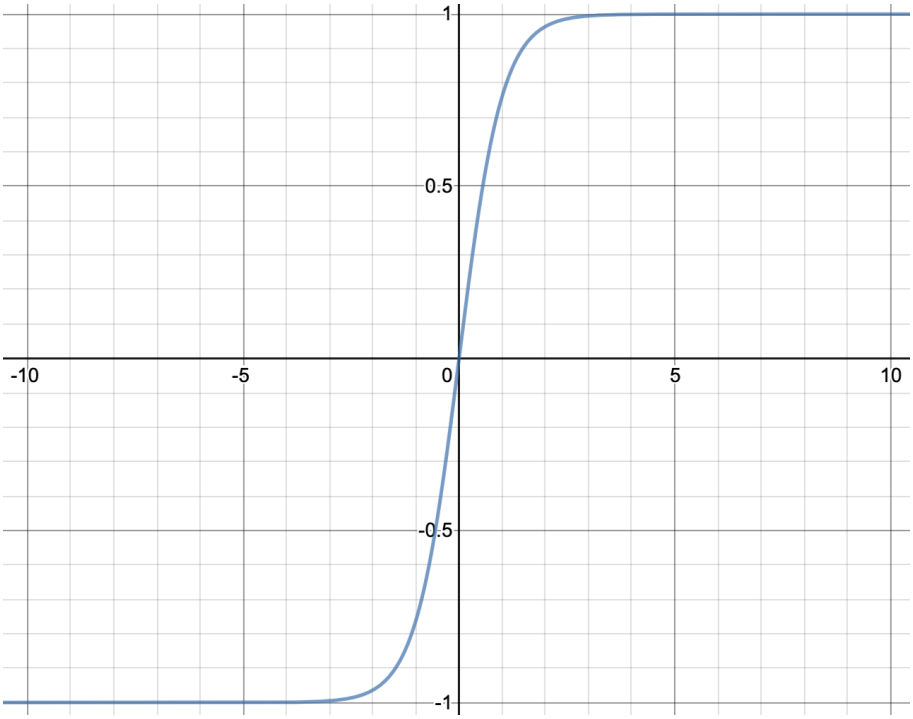


Рисунок 2.1 – Графік активаційної функції $tanh\ (x)$

Цю активаційну функцію почали використовувати через її можливості до запобігання зміщенню та через швидшу збіжність нейронних мереж, що використовують цю функцію, як активаційну, до апроксимованих значень під час тренувань.

Функція Rectified Linear Unit (ReLU), що записується наступним чином:

$$g(x) = \max(0, x) \tag{2.14}$$

демонструє збіжність швидшу, ніж тангенс гіперболічний. Градієнтні методи навчання не можуть отримати інформацію про стан нейронів через те, що вони вихідний сигнал цих нейронів приймає значення 0 на деякому проміжку. Використовуючи ReLU (Рисунок 2.2) як функцію активації, нейрони можуть безповоротно загинути, потрапивши у ситуацію, в якій їм не вдається отримати значення, при якому нейрони знову активуються.

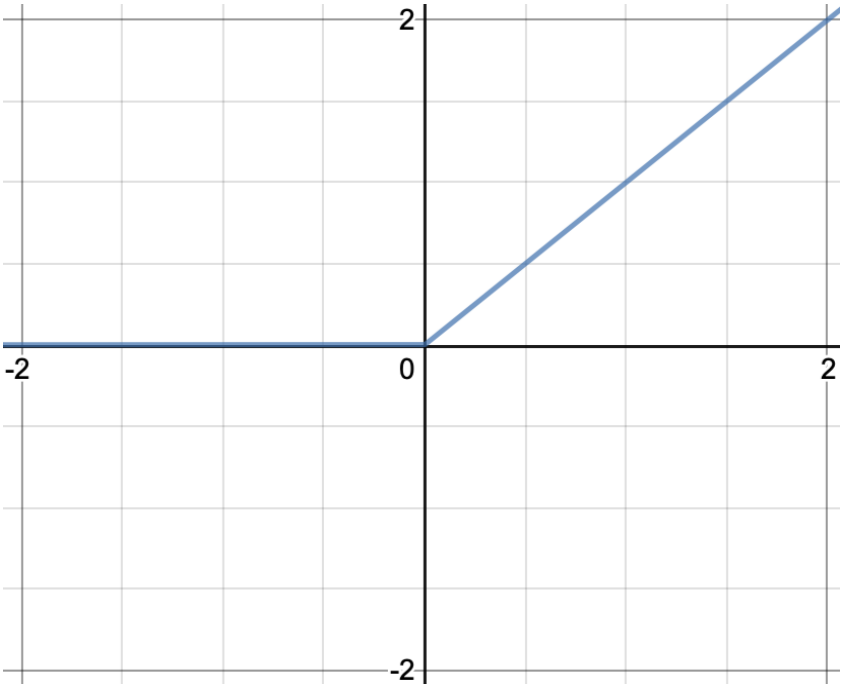


Рисунок 2.2 – графік активаційної функції $ReLU$

2.4.3 Рекурентні нейронні мережі

Ідея архітектури рекурентних нейронних мереж полягає в тому, щоб кожен крок поділяв один і той же прогресуючий (тобто поточний) внутрішній стан, а нейронна мережа мала інформацію не тільки поточний крок часу, але і про

попередній. Така реалізація досягається, наприклад, спільним з'єднанням між прихованими шарами, отриманням сигналу з попереднього кроку часу до прихованих шарів поточного кроку часу. Зв'язок між перерахованими елементами можна формалізувати наступним чином:

$$h(t) = f(h(t-1), x(t), \theta), \tag{2.15}$$

де $f -$ функція прямого поширення;

$h(t) -$ стан у момент часу t;

$x(t) -$ вектор вхідних сигналів;

$\theta -$ вектор параметрів.

Як і більшість нейронних мереж, рекурентні нейронні мережі були описані доволі давно. На початку 1990-х рр. проблема з градієнтом, що зникає, стала основною перешкодою для повторних показників чистої ефективності. Так само, як пряма виражає зміну x поряд із зміною y, градієнт виражає зміну всіх ваг щодо зміни помилки. Якщо ми не можемо знати градієнта, ми не можемо регулювати ваги в напрямку, що зменшить помилку, і мережа перестає вчитися. Проблема зникаючого градієнта виникає через те, що проходячи через шари нейронної мережі в поширюючись в мережі, вектори підпадають під операції множення, що суттєво збільшують або зменшують їх значення, а особливо в рекурентних нейронних мережах. Через те, що компонування зв'язків між шарами нейронних мереж відбувається за допомогою операції перемноження, похідні в цих випадках вразливі до перебільшення значення (так званого «вибуху»), або навпаки, надмірного зменшення значення і зникання. Якщо протрапляємо у ситуацію з надмірним збільшенням градієнту, то такий градієнт називають «вибуховим» і в такому випадку кожна вага буде завеликою і навчання нейронної мережі буде неможливим. Градієнти цих ваг насичуються на найвищому кінці, тобто вони вважаються занадто потужними. Але «вибухові» градієнти можна вирішити відносно легко, оскільки вони можуть бути

усіченими або розсіченими. Також, описана нейронна мережа має суттєвий недолік – вона не має можливості отримати оцінку для подій, що відбувалися достатньо довгий час назад, тобто кажучи у термінах прихованих шарів, вплив прихованих шарів рекурентної нейронної мережі з часом зменшується, що дає нам змогу зробити висновок, що такий тип нейронних можна використовувати для прогнозування і аналізу часових рядів, що не містять в собі великий об’єм даних. Це пояснюється тим, що градієнт функції втрати експоненційно затухає з часом, що було описано вище. Для того, щоб використовувати рекурентну мережу для вирішення поставленої задачі прогнозування і аналізу часових рядів, необхідно зробити модифікацію цієї мережі, щоб мати змогу отримувати дані з прихованих шарів, що мають вплив на поточний внутрішній стан і мати можливість ці дані якимось чином записувати та зберігати, а потім оперувати ними, або робити висновки з приводу цих даних та використовувати отриману інформацію для побудови прогнозу. Рекурентні нейронні мережі, які мають таку модифікацію, називаються нейронними мережами з довгою короткостроковою пам’яттю і позначаються як LSTM.

2.4.4 Нейронні мережі з довгою короткостроковою пам’яттю

Як вже було неодноразово сказано, нейронні мережі з довгою короткостроковою пам’яттю були створені для вирішення проблеми довготривалих залежностей. LSTM були створені в якості модифікації рекурентних нейронних мереж, в яких є можливість зберігати та обробляти інформацію на відносно довгий період часу. Будь-яка рекурентна нейронна мережа має форму ланцюжка повторюваних модулів нейронної мережі. У звичайній RNN структура одного такого модуля проста, наприклад, модуль

може являти собою один шар з функцією активації *tanh* (гіперболічний тангенс), який було розглянуто у минулому розділі, можемо побачити на Рисунку 2.3:

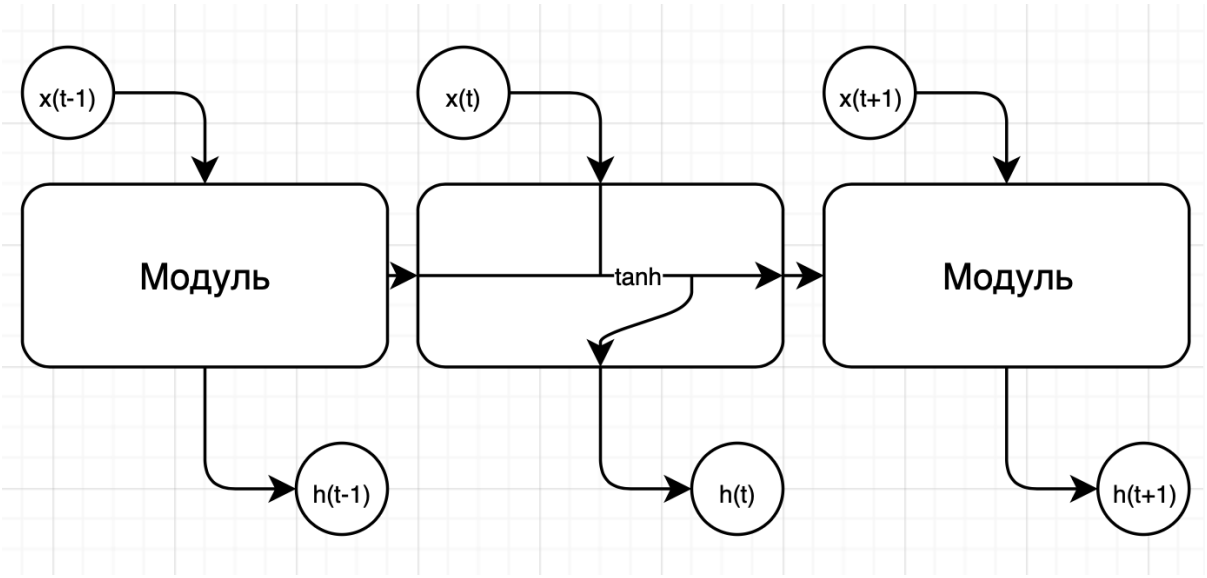


Рисунок 2.3 – Рекурентна нейронна мережа з функцією активації $tanh\ (x)$

Як бачимо, модуль звичайної рекурентної нейронної мережі не є надто складним – в розгортці знаходиться одношаровий перцептрон з вказаною функцією активації. У нейронній мережі LSTM використовується інший, набагато складніший модуль (Рисунок 2.4), який містить в собі декілька функцій активації та багато різноманітних перетворень [14]:

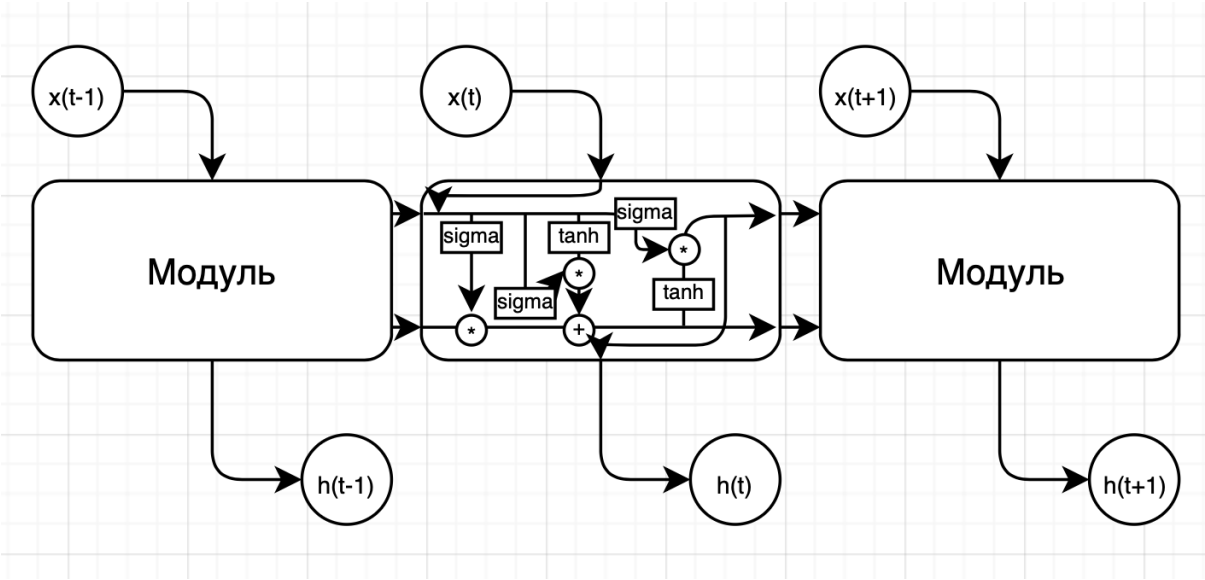


Рисунок 2.4 – Модуль мережі LSTM

На цій схемі маємо наступні умовні позначення (Рисунок 2.5):

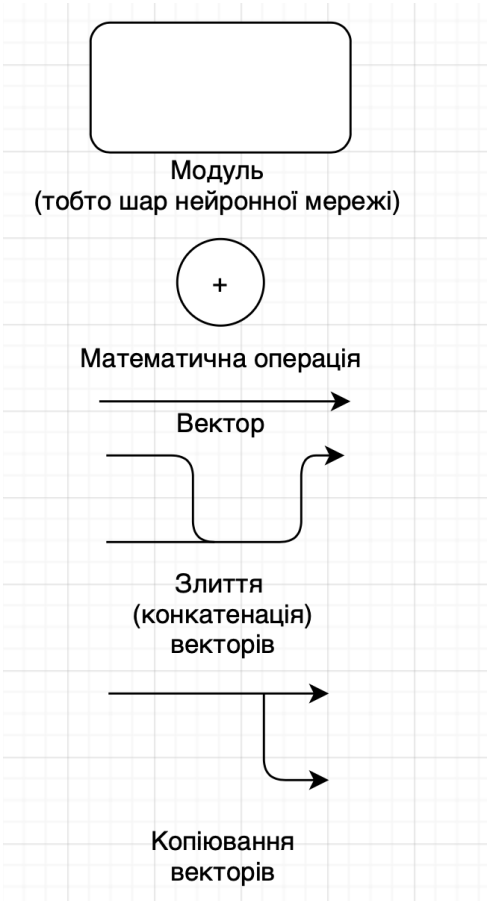


Рисунок 2.5 – Умовні позначення на схемі LSTM

Основна ідея нейронної мережі полягає у зберіганні та передачі «стану клітини» (Рисунок 2.6) – цей так званий стан виглядає як єдина нерозривна лінія, що проходить крізь усі модулі та приймає участь лише у деяких математичних операціях.

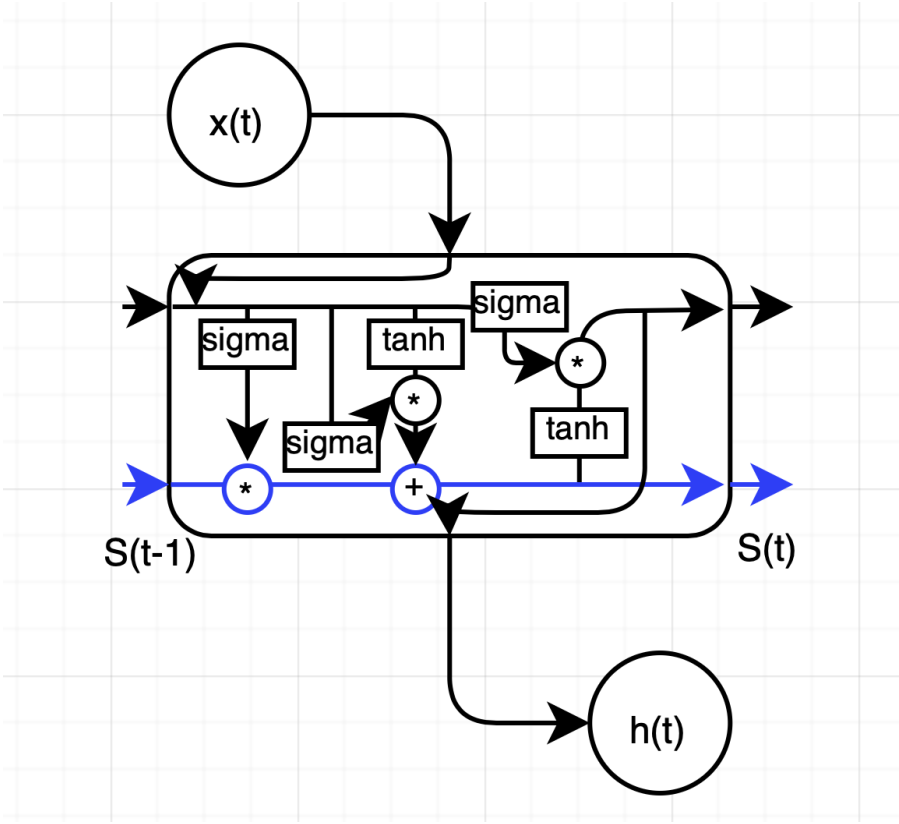


Рисунок 2.6 – «Стан клітини» LSTM

Для видалення інформації з цього неперервного стану використовуються вже описані раніше ворота (gates) та фільтрів, що визначають, яку інформацію необхідно пропускати.

Робота цього модуля починається з прийняття рішення щодо зберігання інформації. Це рішення відбувається на етапі вже описаного раніше фільтра забування (або забуваючого фільтра), робота якого продемонстрована на Рисунку 2.7. Цей фільтр отримує на вхід стан попередньої клітини $h(t-1)$ та вхідний вектор для обраної клітини $\bar{x}(t)$, повертаючи результуючим сигналом число від 0 до 1. Кожне число з «стану клітини» попередньої клітини має пройти

через цей алгоритм. 0 буде позначати, що інформацією потрібно знехтувати повністю, а 1 означає, що всю інформацію необхідно зберегти. Формалізувати такий фільтр можна наступним чином:

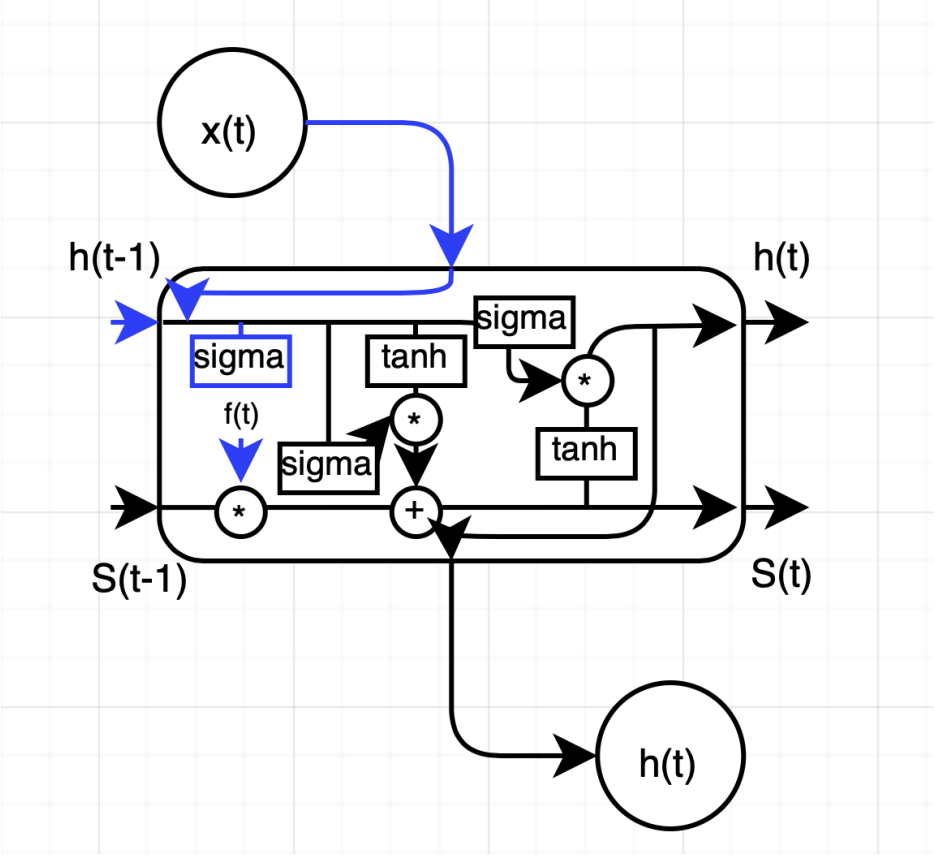


Рисунок 2.7 – Етап забуваючого фільтра

$$f(t) = \sigma\left(w_f \cdot \left(\bar{x}(t), \bar{h}(t-1)\right) + b_f\right), \tag{2.16}$$

де $f(t) \in [0,1]$ – результуючий сигнал фільтру забування;

w_f – значення вагів даного фільтру;

b_f – зміщення фільтру.

Наступний етап полягає у прийнятті рішення щодо зберігання нової інформації у описаному «стані». На початку розпізнавання використовуємо етап вхідного фільтра, на якому визначається, які саме числа підлягають оновленню. Цей процес можна описати наступним чином, використовуючи ті ж позначення,

що і раніше, за винятком індексів, що зараз репрезентують проміжний стан сигмоїдального шару (той самий шар, що повертає значення від 0 до 1 в якості результуючого сигналу):

$$\mathrm{j(t)} = \sigma\big(\mathrm{w_j} \cdot \big(\mathrm{x(t), h(t-1)}\big) + \mathrm{b_j}\big), \tag{2.17}$$

де $\mathrm{w_j}$ – значення вагів проміжного стану сигмоїдального шару;

$\mathrm{b_j}$ – зміщення проміжного стану сигмоїдального шару.

Після роботи тимчасового шару, шар $tanh\ (x)$ повертає набір даних, які теоретично можуть поповнити «стан» (Рисунок 2.8). Значення цього набору записуються наступним чином, використовуючи ті ж позначення, що і раніше, за винятком індексів, що в даному випадку позначають безпосередньо проміжний «стан клітини»:

$$\tilde{\mathrm{S}}\mathrm{(t)} = \tanh\big(\mathrm{w_S} \cdot \big(\mathrm{x(t), h(t-1)}\big) + \mathrm{b_S}\big), \tag{2.18}$$

де $\mathrm{w_S}$ – значення вагів проміжний «стану клітини»;

$\mathrm{b_S}$ – зміщення проміжний «стану клітини».

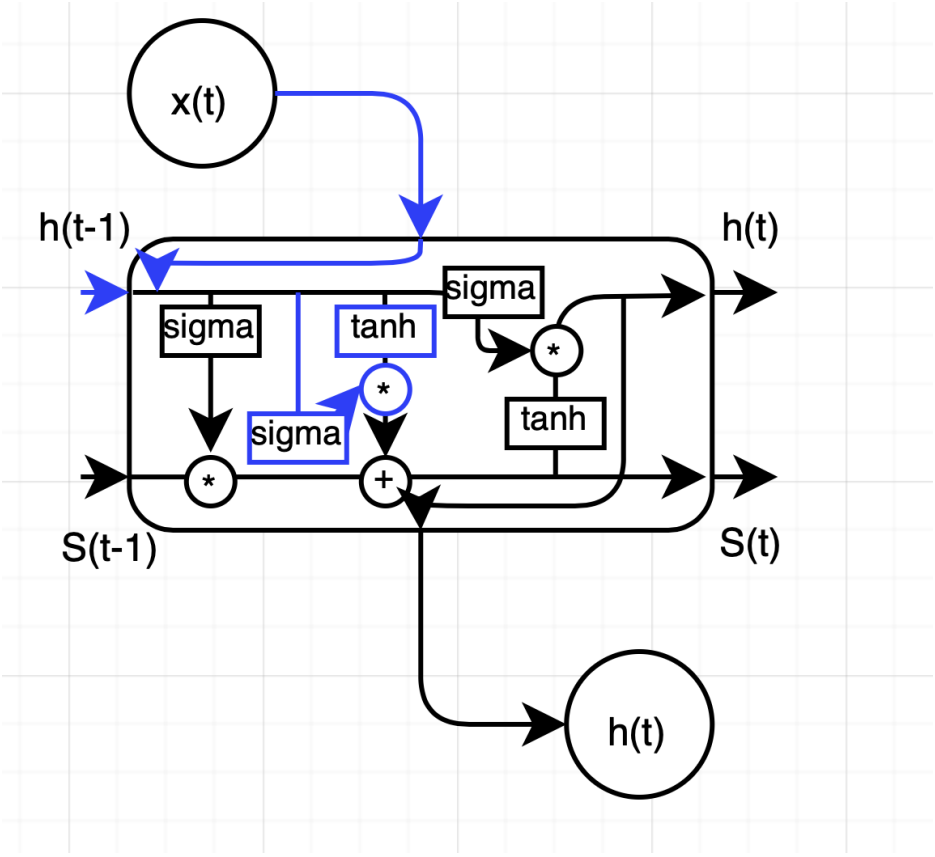


Рисунок 2.8 – Етап вхідного фільтра та шару $tanh\ (x)$

Наступний етап відбувається фактично за допомогою фільтрів, що були описані вище, тобто забуваючого та вхідного фільтрів. Завдяки описаним вище фільтрам ми можемо формалізувати задачу отримання нового стану клітини:

$$S(t) = f(t) \cdot S(t-1) + j(t) \cdot \tilde{S}(t) \tag{2.19}$$

Отриману формулу можна представити у вигляді рисунку (Рисунок 2.9):

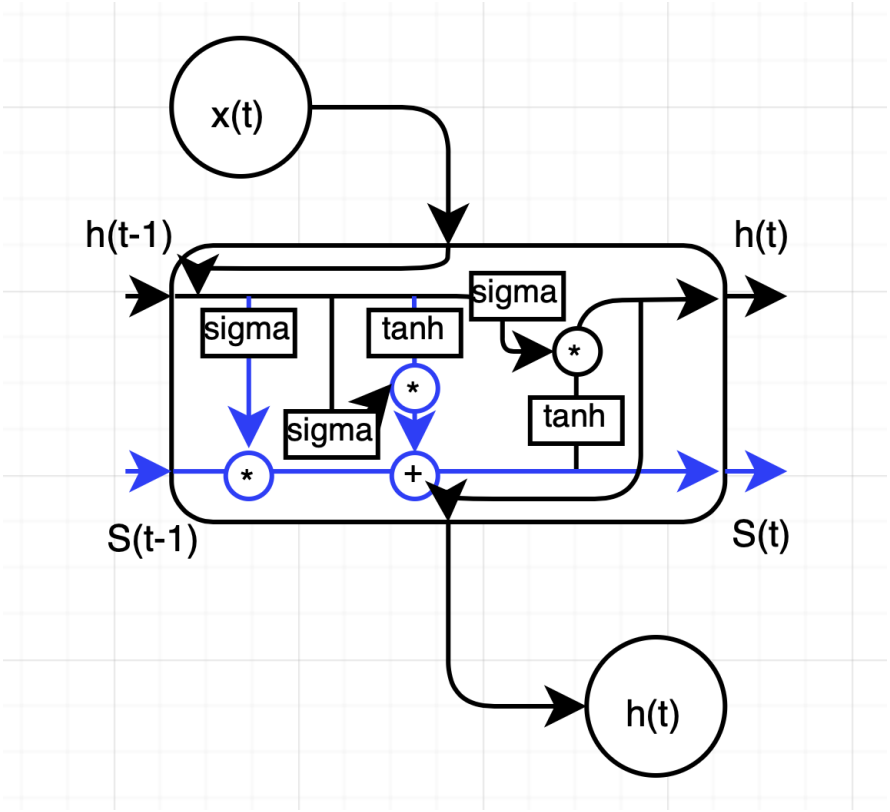


Рисунок 2.9 – Етап перенесення інформації у новий «стан клітини»

Після отримання нового «стану клітини» маємо прийняти рішення стосовно вихідного сигналу цього модуля. Вихідні дані будуть засновані на нашому «стані клітини», але вони будуть опрацьовані деякими фільтрами. Цей етап буде реалізовано за допомогою вихідного фільтра, детальний алгоритм роботи якого зараз буде наведено. Спочатку ми застосовується сигмоїдальний шар, який вирішує, яку інформацію зі «стану клітини» необхідно буде вивести. Дана процедура може бути формалізована наступним чином:

$$g(t) = \sigma(w_g \cdot (x(t), h(t-1)) + b_g) \qquad (2.20)$$

Потім значення «стану клітини» проходять через $tanh-$ шар, щоб отримати на виході значення з діапазону від -1 до 1, і перемножуються з вихідними значеннями сигмоїдального шару, що дозволяє виводити тільки

необхідну інформацію (Рисунок 2.10). Формалізація цієї процедури виглядає наступним чином:

$$h(t) = g(t) \cdot \tanh(S(t)) \tag{2.21}$$

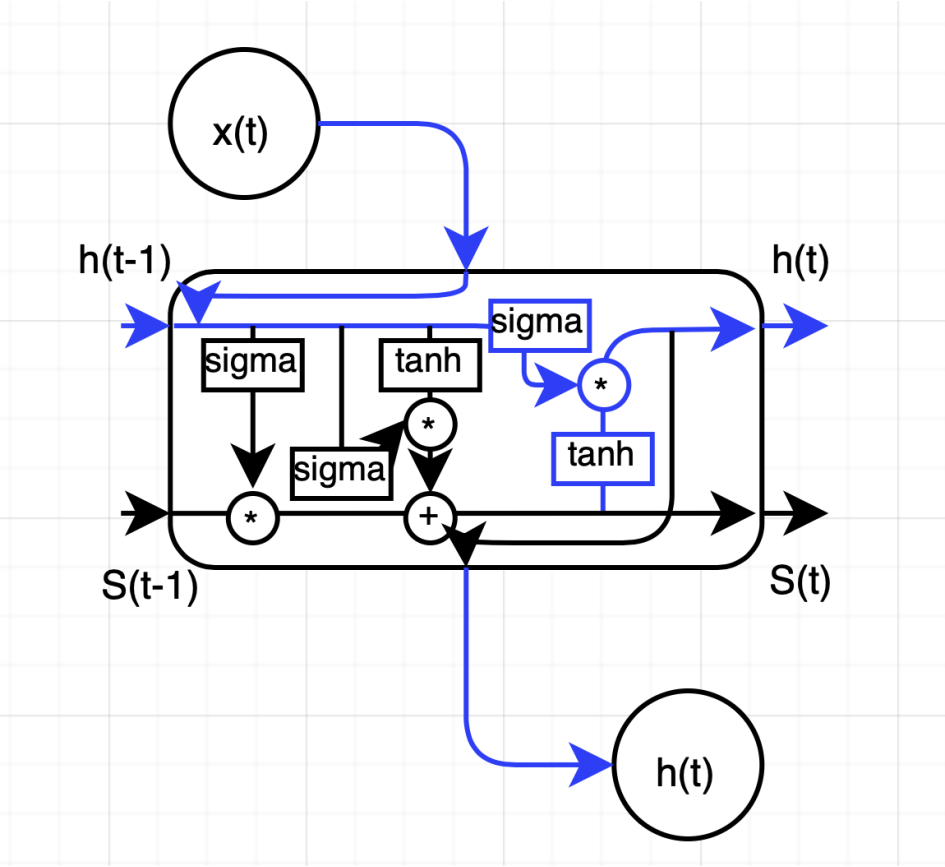


Рисунок 2.10 – Етап перенесення інформації у вихідний сигнал

Після виконання даного етапу відбувається перехід на наступну ітерацію, тобто в даному випадку перехід до наступного «уявного» модуля, уявного через те, що розглядається рекурентна нейронна мережа, яка насправді замість ланцюга модулів має змогу повертати свій стан на вхід до одної й тої ж клітини. Таким чином було розглянуто базовий алгоритм роботи нейронної мережі з довгою короткостроковою пам'яттю. Також були продемонстровані ключові елементи даної нейронної мережі, а саме поняття модулів (клітин), «клітинного стану», різноманітних фільтрів (маються на увазі забуваючий, вхідний та

вихідний), її можливості та особливості передачі інформації та реалізацію передачі даної інформації.

2.5 Висновки до розділу

Було визначено та розглянуто основні статистичні моделі та принципи моделювання а також дослідивши роботу рекурентних нейронних мереж, а саме нейронних мереж з довгою короткостроковою пам’яттю. Як можемо бачити, проблеми втрачання взаємозв’язку між подіями, що відбувалися через деякий проміжок часу (що є абсолютно реальною задачею для даних, що не було згенеровано або змодельовано спеціально для вирішення поставленої задачі прогнозування, а для реальних даних, які необхідно буде моделювати програмним забезпеченням), вирішуються за допомогою спеціальної різновидності рекурентних нейронних мереж – нейронних мереж з довгою короткостроковою пам’яттю. Було розглянуто алгоритм роботи даних рекурентних нейронних мереж, основні принципи та алгоритми збереження, перетворення та передачі даних у рекурентних модулях цих мереж.

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ПОБУДОВИ РЕКУРЕНТНИХ НЕЙРОННИХ МЕРЕЖ З ДОВГОЮ КОРОТКОСТРОКОВОЮ ПАМ'ЯТТЮ

3.1 Вибір даних для моделювання

Для проведення моделювання був обраний унікальний датасет даних, розмічений за координатами басейну річки Десна. Даний датасет (або банк даних) має стовпчики з датами, починаючи від 1 Січня 1979 року, а також з гідрометеорологічними замірами, такими як мінімальна температура повітря, середня температура повітря, максимальна температура повітря, кількість опадів за день, вологість, швидкість вітру, хмарність, а також кількість сонячної радіації та середня кількість снігу за день. Приклад даних у датасеті можемо побачити на Рисунку 3.1:

date	t_aver	t_min	t_max	pcp	hmd	wnd	cloud	slr	snow_aver
1979-01-01	-14,1613	-14,3937	-9,80341	0,641792	81,55762	3,98387	1	5,067189	85,08688
1979-01-02	-16,0076	-19,3654	-12,9741	4,979034	78,48497	5,002826	1	5,383169	90,37296
1979-01-03	-17,8547	-21,1729	-17,6829	1,515452	78,23332	3,038331	0,890867	11,43886	100,0664
1979-01-04	-14,6853	-18,5256	-12,8769	0,736598	82,30986	3,476199	0,928638	10,11986	102,7002
1979-01-05	-16,7162	-18,4732	-15,0623	1,311574	76,7992	5,72576	0,981035	9,53597	104,4528
1979-01-06	-14,9236	-17,0002	-13,4395	1,021919	80,15249	5,380274	0,946124	11,32418	108,1898
1979-01-07	-16,6116	-17,5956	-15,7094	0,06501	79,70879	3,744529	0,964041	12,40683	109,0435
1979-01-08	-8,28855	-15,6397	-3,04716	2,932306	78,3811	6,048417	0,997702	6,04317	112,1371
1979-01-09	-2,37465	-4,59574	-1,53811	0,662198	79,24575	3,626678	0,964414	8,123827	116,3988
1979-01-10	-2,68419	-6,35382	-0,36721	1,314283	85,00565	3,662828	0,952619	7,36768	117,9916
1979-01-11	1,120704	-0,78188	2,272909	0,4585	96,35706	3,392795	0,938624	8,994148	118,9427
1979-01-12	1,836734	1,86154	3,540981	1,14002	99,36181	5,417136	1	6,614036	107,2984
1979-01-13	2,825777	0,703709	4,104568	11,74421	91,90984	6,465888	0,999402	5,909968	82,83849
1979-01-14	0,454794	-0,10309	1,452792	2,84039	93,67376	5,970146	0,985207	7,687665	78,9153
1979-01-15	-1,99363	-5,39187	0,720129	5,692337	91,76841	5,898274	0,998487	7,853918	83,61512
1979-01-16	-4,02616	-5,52666	-2,86254	0,969369	88,09979	4,996929	0,997285	10,75382	89,89307
1979-01-17	-6,11154	-6,73201	-4,42504	1,013793	86,07686	4,147983	0,995153	11,42934	92,05696
1979-01-18	-6,00265	-8,57639	-4,74764	0,089389	86,85887	3,175094	0,96238	13,46443	92,72026
1979-01-19	-5,86577	-8,82658	-5,76793	0,194668	87,33918	4,362787	0,886534	13,35126	92,71604
1979-01-20	-6,33439	-7,89857	-6,0128	0,296698	86,20019	3,346088	0,969718	12,77338	92,95711
1979-01-21	-9,57609	-11,5408	-7,75214	0,101307	81,9496	3,809457	0,94713	18,04037	93,07317
1979-01-22	-12,2892	-12,4358	-10,5885	0,056522	79,05168	3,917906	0,938124	17,25869	93,01725
1979-01-23	-9,59325	-12,2652	-7,50728	1,994178	84,54059	3,629803	0,980463	11,98118	95,70801
1979-01-24	-8,60648	-10,2773	-5,42589	0,099501	84,63348	4,561782	0,907231	20,24272	97,59405
1979-01-25	-0,6059	-5,27648	1,934757	9,777846	95,46419	5,527149	0,99925	6,327101	101,5207
1979-01-26	-0,9067	-3,28373	1,389631	1,502631	95,13669	2,910866	0,999749	12,65721	104,8315
1979-01-27	0,421927	-0,94964	2,046502	0,175527	95,75419	2,80447	0,943255	19,47856	101,2372
1979-01-28	0,787455	-0,72273	2,803458	2,286542	93,95943	5,466804	0,996487	18,56918	96,87532
1979-01-29	2,251459	1,174018	3,446725	0,425092	96,25441	2,837217	0,988745	11,63365	82,43911
1979-01-30	1,609139	-1,28613	3,714915	8,914298	93,708	5,628169	0,999617	4,410693	71,66538
1979-01-31	0,318497	-1,86796	1,164197	0,900387	76,60214	6,147501	0,960811	22,079	67,93729

Рисунок 3.1 – Приклад даних з датасету

На даному прикладі наведено всі перераховані метрики за перший місяць 1979 року. Для тестових прикладів моделювання був обраний проміжок з 9 листопада 2018 року по 29 квітня 2019 року, за дані були взяті значення сонячної радіації, що вимірюються у $\frac{Дж}{м^2}$. Слід помітити, що у програмній реалізації наявна можливість використовувати сам банк даних, але через велику кількість записів було прийняте рішення отримувати частини для моделювання власноруч та проводити моделювання вже з готовими та опрацьованими даними.

Для демонстрації роботи деяких окремих конфігурацій нейронної мережі було використано дані про ціни фінансових акцій міжнародної компанії на момент закриття торгівлі в кінці дня. Приклад можемо побачити на Рисунку 3.2:

2019-05-31	67.469997
2019-06-03	67.580001
2019-06-04	70.499998
2019-06-05	69.299999
2019-06-06	70.700003
2019-06-07	75.07
2019-06-10	77.239998
2019-06-11	78.04
2019-06-12	75.93
2019-06-13	75.789997
2019-06-14	73.27
2019-06-17	74.139997
2019-06-18	76.86
2019-06-19	70.260001
2019-06-20	71.719999
2019-06-21	69.509999
2019-06-24	70.920003
2019-06-25	68.809997
2019-06-26	70.029998
2019-06-27	70.030001
2019-06-28	69.240002
2019-07-01	73.709999
2019-07-02	71.949998

Рисунок 3.2 – Приклад даних з датасету

3.2 Вибір інструментів

У наші часи існує величезна кількість різноманітних готових програмних рішень, які дають можливість реалізувати моделі для прогнозування часових рядів. Деякі готові програмні продукти було розглянуто у першому розділі, тому перейдемо до огляду компонентів розробки, з яких зазвичай складаються програмні продукти, що реалізовують побудови математичних моделей. Використання мови Python для програмної реалізації обумовлено великою кількістю запропонованих моделей та бібліотек, які користувач може використовувати при моделюванні та побудові різноманітних процесів, в перелік яких входять як нейронні мережі, так і звичайні математичні моделі. Використання таких типів бібліотек значно розширює можливості дослідника на отримання різноманітних метрик та результатів моделювання, адже вбудовані бібліотеки оптимізовані для виконання моделювання та презентування результатів моделювання користувачу. Деякі бібліотеки використовуються для графічного представлення моделей, деякі ж використовуються для обробки масивів даних та обрахунків. Наприклад, бібліотека Scikit-learn. До завдань бібліотеки не входить завантаження, обробка, маніпуляція даними і їх візуалізація. Scikit-learn спеціалізується на алгоритмах машинного навчання для вирішення завдань навчання з учителем: класифікації і регресії, а також для задач навчання без учителя: кластеризації (розбиття даних по класах , які модель визначить сама), зниження розмірності (подання даних в просторі меншої розмірності з мінімальними втратами корисної інформації) і детектування аномалій. В нашому випадку, бібліотека sklearn використовується для побудови та оцінки метрик MSE та RMSE, що були описані раніше. Також була використана бібліотека NumPy – головна бібліотека для Python, якщо мова йде про математику. В Python бібліотека Numpy незамінна для роботи з числовими масивами, векторами і матрицями, а також дозволяє будувати графіки і

гістограми. Для ефективної обробки датасетів (тобто банків даних) використовується бібліотека pandas. Pandas – високорівнева бібліотека, написана за допомогою бібліотеки NumPy. Ця бібліотека надає можливість ефективно використовувати вхідні дані, зчитувати інформацію з файлів, перетворювати інформацію на масиви, словники та часові ряди. Оскільки використаний для збору даних файл є набором даних у вигляді матриці, ця бібліотека якнайкращим чином дозволяє обробляти дані та маніпулювати ними. Була використана бібліотека keras – бібліотека, що використовується для обрахунків та машинного навчання за допомогою різних видів програмних бібліотек. У нашому випадку бібліотека обрахунків це TensorFlow. TensorFlow – бібліотека, шо була розроблена та створена компанією Google для моделювання та навчання нейронних мереж. За допомогою цих бібліотек навчання нейронних мереж стало доступним для широкого кола дослідників. Для перевірки результатів моделювання та альтернативного моделювання за допомогою статичтиної моделі використовується бібліотека statsmodels, що забезпечує класи та функції для оцінки багатьох різних статистичних моделей, а також для проведення статистичних тестів та вивчення статистичних даних. Для виконання основних математичних операцій використовується бібліотека math, що надає великий функціонал роботи з числами. Для графічного представлення результатів моделювання використовується бібліотека matplotlib.

3.3 Побудова моделей нейронних мереж та порівняння з статистичною моделлю

Було виконано моделювання на прикладі вказаних раніше даних за допомогою моделі ARIMA та побудови нейронної мережі з довгою короткостроковою пам’яттю, порівняння моделей відбувається за параметрами

MAE, RMSE та DW, що були описані раніше. Для моделі нейронної мережі також використовується метрика MAPE, для оцінки прогнозу тестових даних. Для початку на Рисунку 3.3 продемонструємо, як виглядає графік реальних даних:

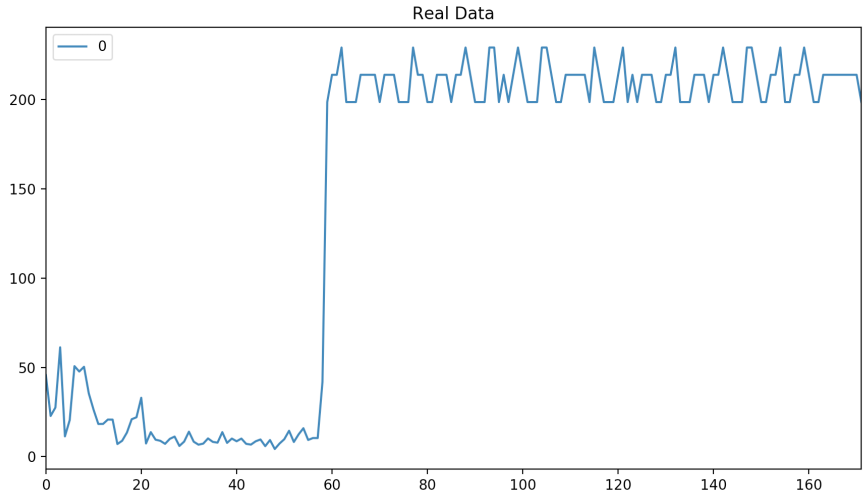


Рисунок 3.3 – Графік реальних даних

Модель ARIMA використовується з коефіцієнтами ARIMA(2,2,2) з подальшим пристосуванням моделі до більш реальних результатів за допомогою методу .fit(). Моделювання за допомогою математичної моделі дало наступні результати, графіки похибок яких продемонстровано на Рисунку 3.4:

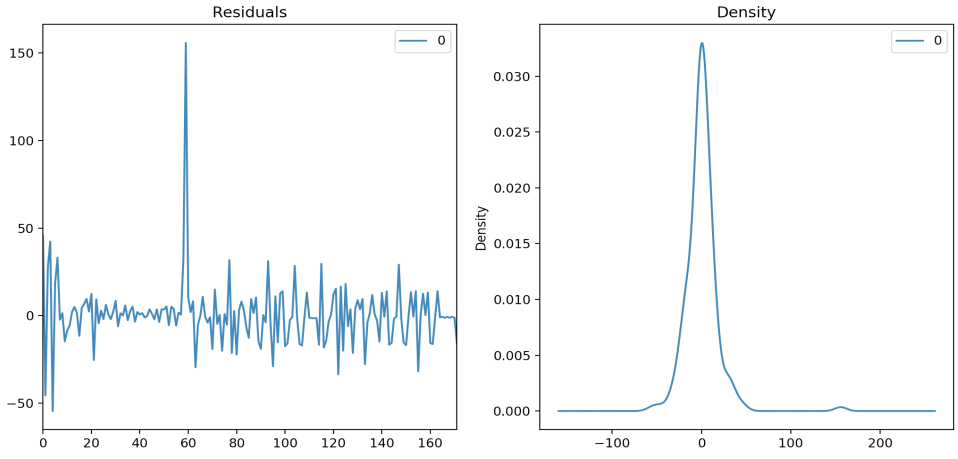


Рисунок 3.4 – Похибки моделювання за допомогою математичної моделі

На даних графіках продемонстровано похибки моделі ARIMA а також ядрову оцінку густини розподілу похибок. Метрики для даної моделі приймають наступні значення: $\mathrm{RMSE} = 18.5998, \mathrm{MAE} = 10.6676, \mathrm{DW} = 2.0738$.

Ці дані свідчать про те, що у похибок відсутня автокореляція, що дає змогу вказати, що модель є досить точною. Але значення похибок для даної моделі перевищують значення для моделі, побудованої за допомогою нейронної мережі з довгою короткостроковою пам'яттю, графіки розподілу похибок якої продемонстровано на Рисунку 3.5.

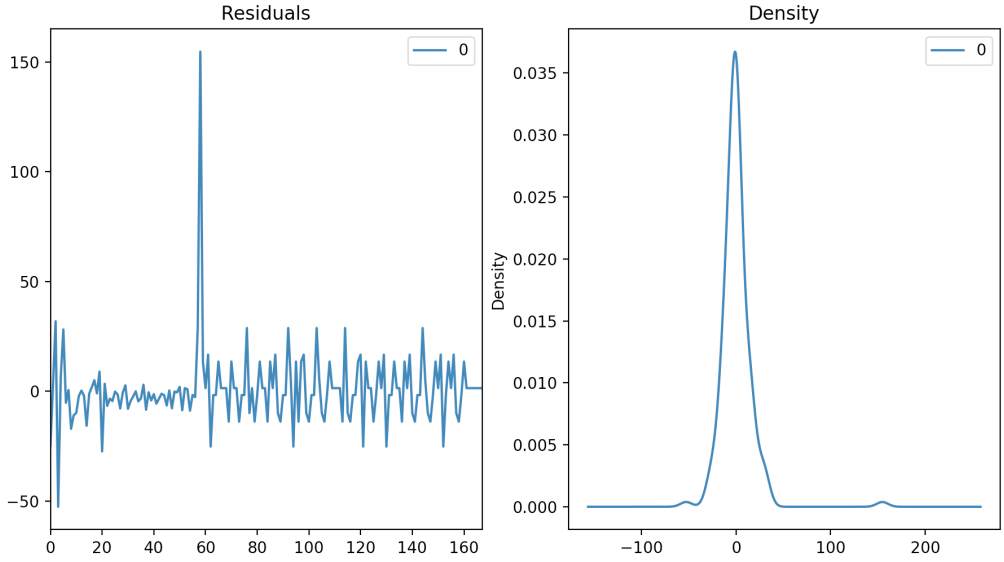


Рисунок 3.5 – Похибки моделювання за допомогою нейронної мережі

В даному випадку нейронна мережа складалась з вхідного шару з функцією активації $ReLU$, LSTM з функцією активації $tanh$ та фітнес-функцією МНК, метод пошуку розв'язку – градієнтний спуск. Навчання відбувалося за 100 епох. Для даної моделі значення метрик для прогнозованих даних, що були отримані під час прогнозування, наступні: $\mathrm{RMSE} = 11.2020, \mathrm{MAE} = 8.1408, \mathrm{DW} = 2.2264, \mathrm{MAPE} = 0.0385$.

Ці дані свідчать про менші похибки, але підрахований коефіцієнт DW вказує на наявність автокореляції у залишках моделі. Результати прогнозування продемонстровано на Рисунку 3.6:

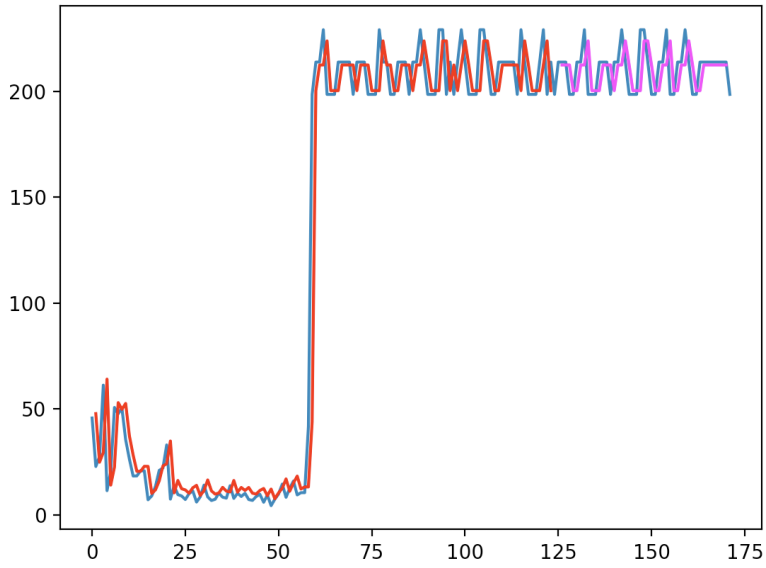


Рисунок 3.6 – Результати прогнозування за допомогою нейронної мережі

На даному рисунку синім кольором позначені реальні дані, червоним кольором позначаються результати тренувальних даних, а рожевим – результати тестового прогнозу, тобто спрогнозовані дані. Спробуємо тепер змінити кількість епох для навчання мережі та підвищити їх кількість до 300, при цьому всі інші характеристики нейронної мережі (тобто кількість шарів, активаційні функції, фітнес функцію та метод пошуку розв’язку залишимо без змін). Отримаємо наступні результати, графіки похибок яких продемонстровано на Рисунку 3.7:

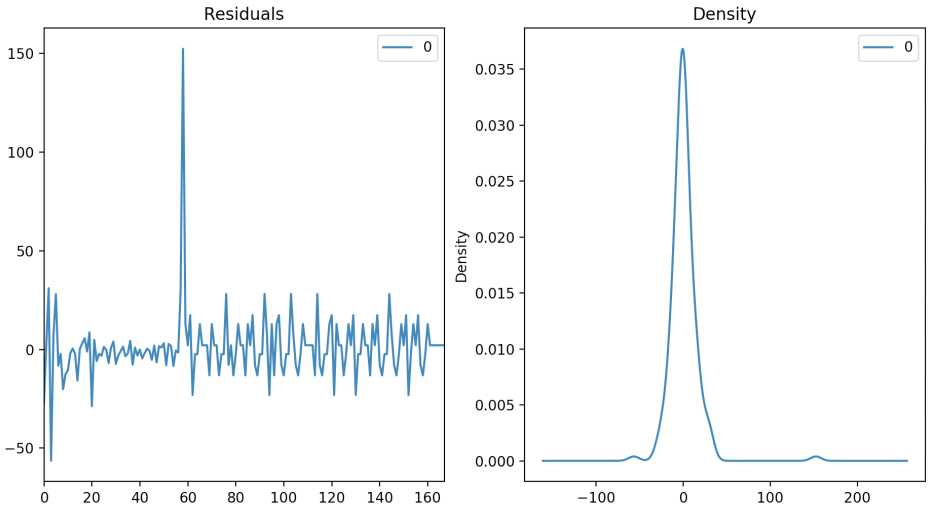


Рисунок 3.7 – Похибки моделювання за допомогою нейронної мережі при збільшенні кількості епох до 300

В цьому випадку отримаємо наступні метрики: $\mathrm{RMSE} = 10.8351, \mathrm{MAE} = 8.1947, \mathrm{DW} = 2.1760, \mathrm{MAPE} = 0.0386$. Як можемо бачити, статистика DW стала ближчою до 2х, що дає зробити висновок, що при більш довгому навчанні моделі автокореляція у залишках зникає. Результати моделювання продемонстровано на Рисунку 3.8:

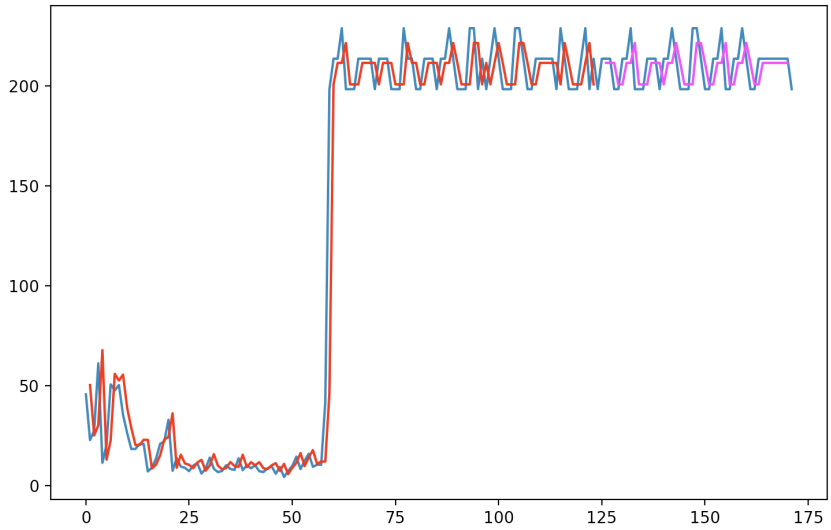


Рисунок 3.8 – Результати прогнозування за допомогою нейронної мережі при збільшенні кількості епох до 300

На даному рисунку всі позначення з минулого результату прогнозування залишаються.

При збільшенні кількості епох до 1000 та при тих самих характеристиках мережі отримуємо наступні результати, графіки похибок яких продемонстровано на Рисунку 3.9:

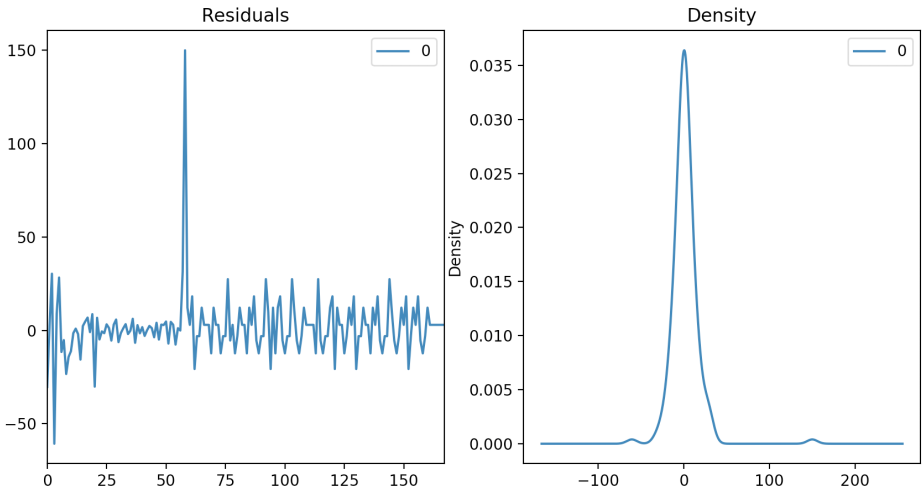


Рисунок 3.9 – Похибки моделювання за допомогою нейронної мережі при збільшенні кількості епох до 1000

В цьому випадку отримаємо наступні значення метрик: $\mathrm{RMSE} = 10.5121, \mathrm{MAE} = 8.2517, \mathrm{DW} = 2.0901, \mathrm{MAPE} = 0.0389$. Як бачимо, при збільшенні кількості епох та відповідно при більшому часі навчання мережі, отримуємо кращі результати, про які свідчить про менша помилка та наближення коефіцієнту DW до 2х. Результати прогнозування продемонстровано на Рисунку 3.10:

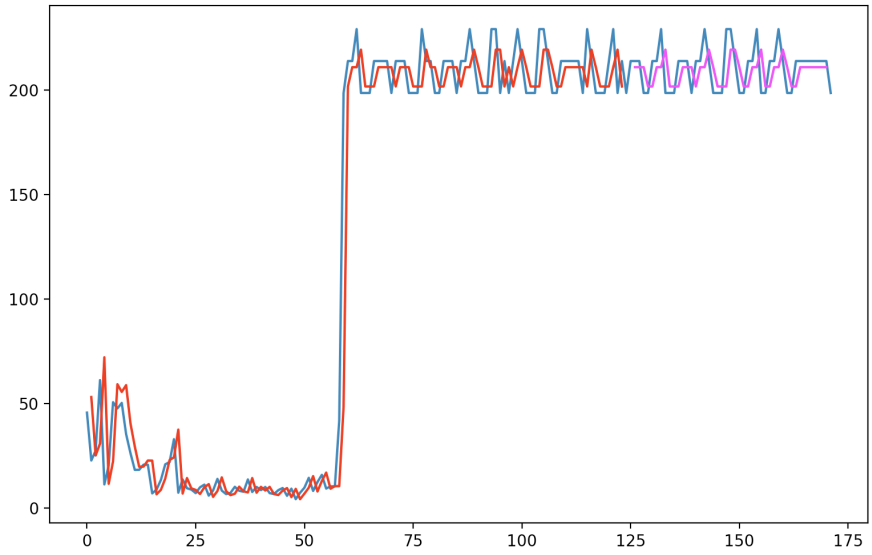


Рисунок 3.10 – Результати прогнозування за допомогою нейронної мережі при збільшенні кількості епох до 1000

На даному рисунку всі позначення з минулого результату прогнозування залишаються.

Для подальшого дослідження використаємо другий набір даних. Для початку на Рисунку 3.11 продемонструємо, як виглядає графік реальних даних:

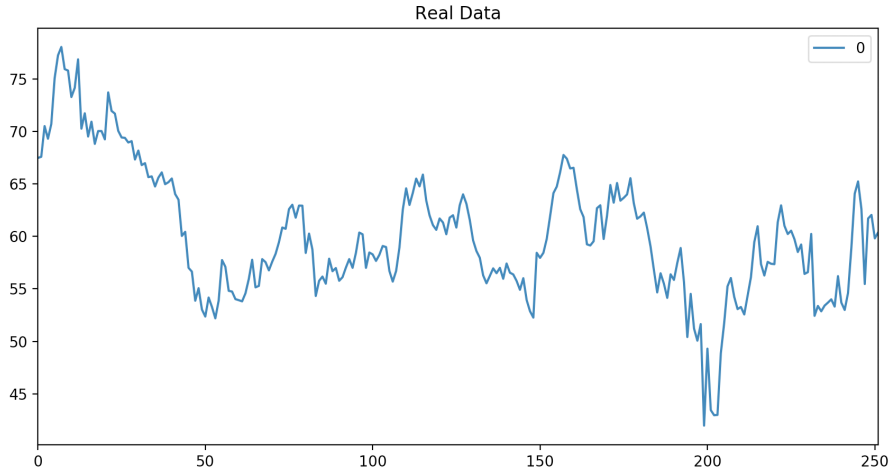


Рисунок 3.11 – Графік реальних даних

Виконаємо моделювання за допомогою математичної моделі. Графіки похибок продемонстровано на Рисунку 3.12:

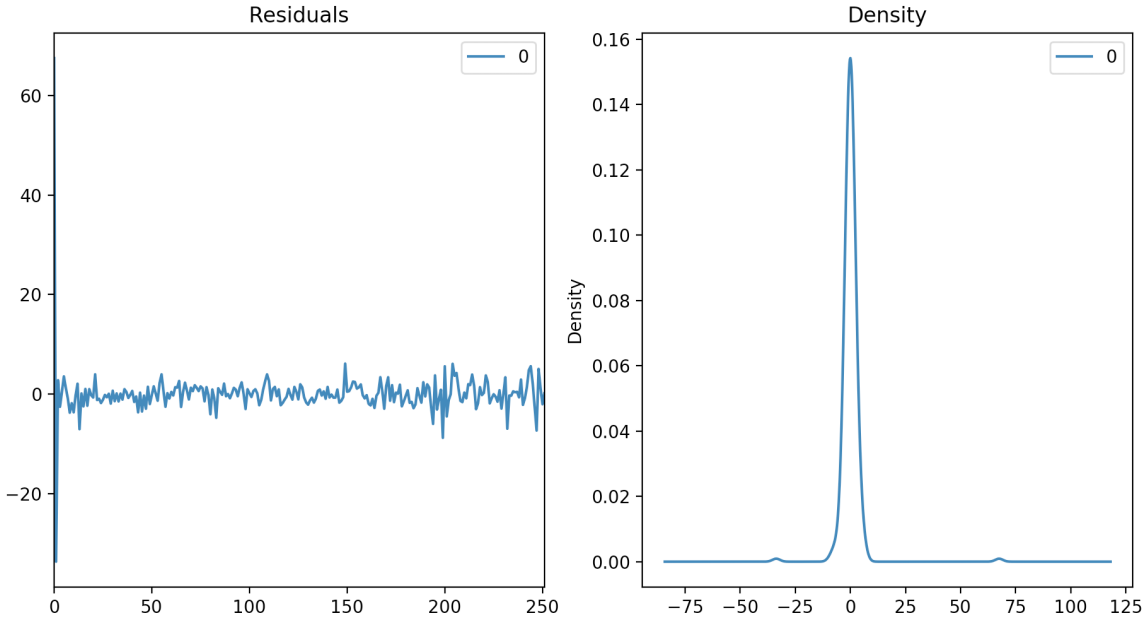


Рисунок 3.12 – Похибки моделювання за допомогою математичної моделі

Метрики для даної моделі приймають наступні значення: $\mathrm{RMSE} = 5.2180, \mathrm{MAE} = 1.9961, \mathrm{DW} = 2.0267$. Як можна помітити, похибки і метрики суттєво покращились на даній вибірці. Перейдемо до моделювання за

допомогою нейронної мережі. Дещо змінимо конфігурацію нейронної мережі – змінимо активаційну функцію LSTM шару на ReLU, метод оптимізації змінимо на метод стохастичного градієнту та додамо моделі можливість мати короткострокову пам’ять, визначивши, що шар тепер зможе працювати з 1ю вибіркою перед переходом до наступної епохи навчання. Для кількості епох 100 маємо наступні результати, графіки похибок яких продемонстровано на Рисунку 3.13:

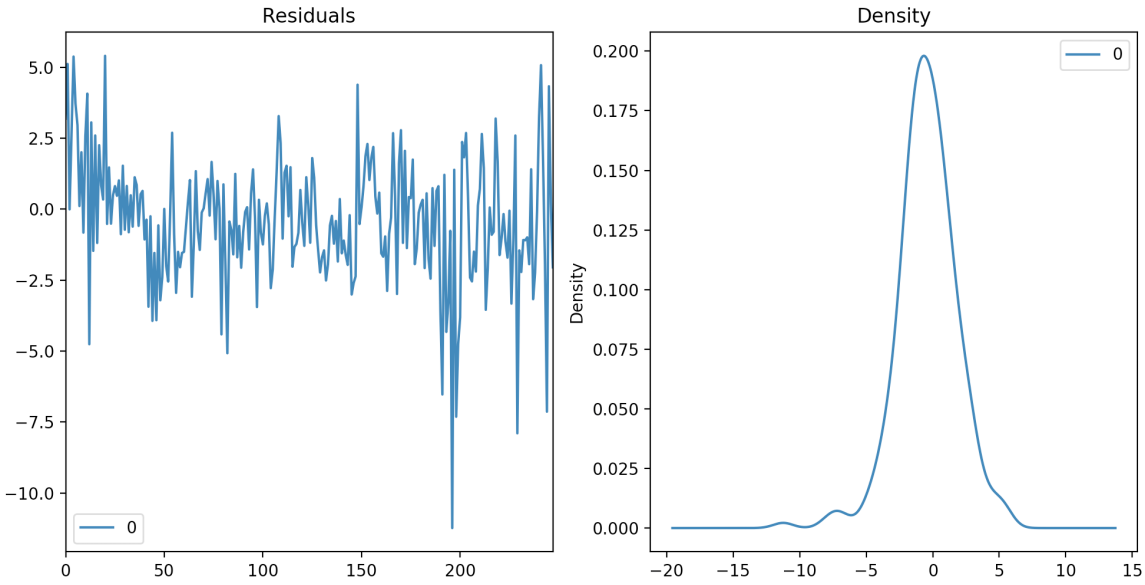


Рисунок 3.13 – Похибки моделювання за допомогою нейронної мережі з пам’яттю

В цьому випадку отримаємо наступні значення метрик: $\mathrm{RMSE} = 3.0838, \mathrm{MAE} = 2.2987, \mathrm{DW} = 1.7067, \mathrm{MAPE} = 0.04355$. Як бачимо, для мережі з пам’яттю метрики погіршились. Це може свідчити про те, що мережі було надано недостатньо часу для навчання. Результати прогнозування продемонстровано на Рисунку 3.14:

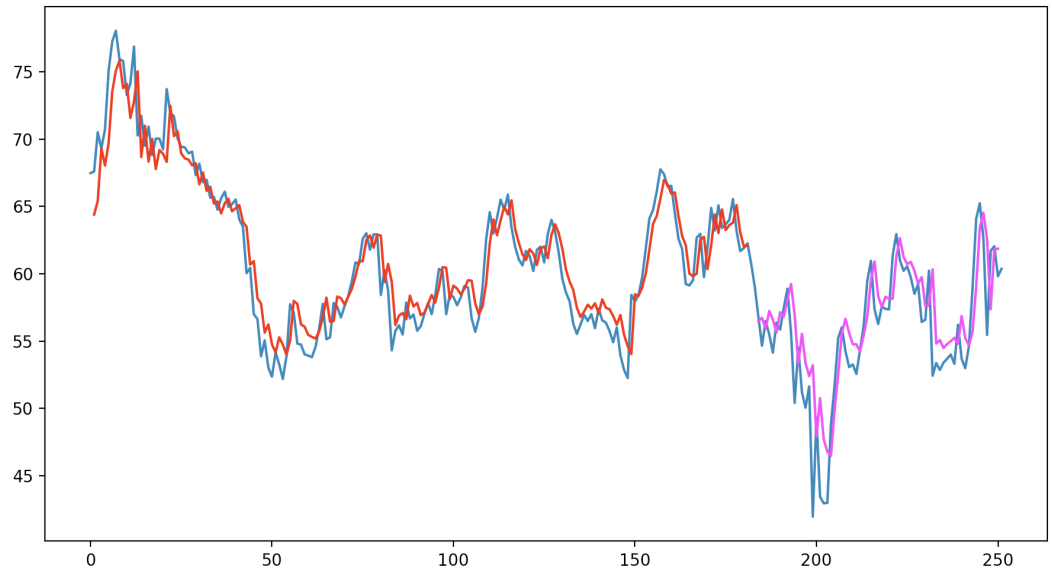


Рисунок 3.14 – Результати прогнозування за допомогою нейронної мережі з пам’яттю

Спробуємо значно збільшити кількість епох при збереженні характеристик мережі. При кількості епох рівній 500 отримаємо наступні результати, графіки похибок яких продемонстровано на Рисунку 3.15:

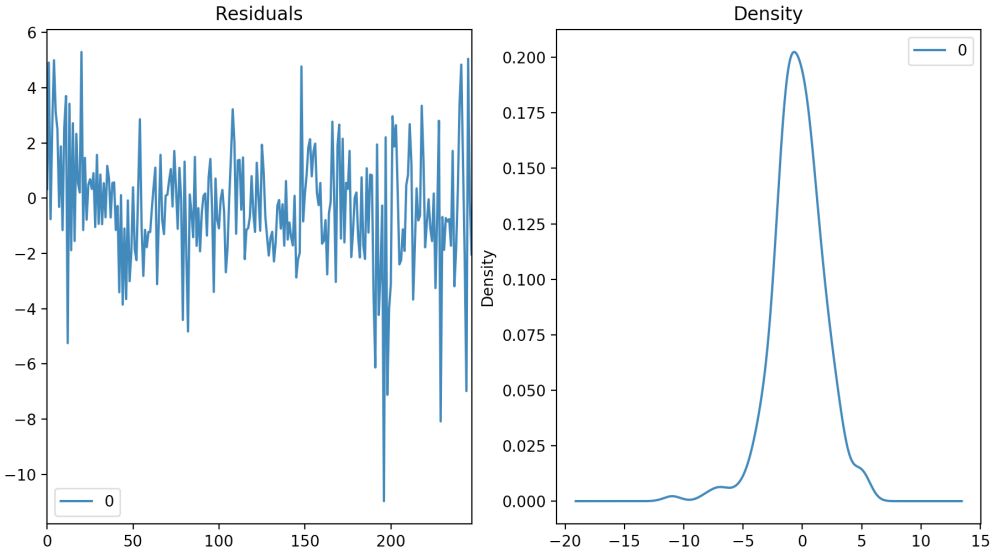


Рисунок 3.15 – Похибки моделювання за допомогою нейронної мережі з пам’яттю при кількості епох 500

В цьому випадку отримаємо наступні значення метрик: $\mathrm{RMSE} = 3.0247, \mathrm{MAE} = 2.2530, \mathrm{DW} = 1.9202, \mathrm{MAPE} = 0.0425$. Як можемо бачити, статистичні показники покращились. На Рисунку 3.16 продемонстровано результат прогнозування:

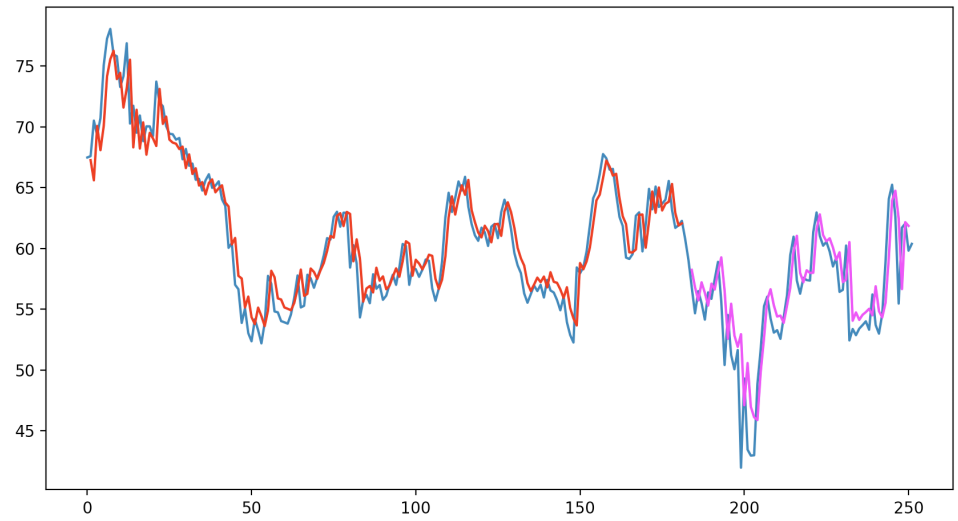


Рисунок 3.16 – Результати прогнозування за допомогою нейронної мережі з пам’яттю при кількості епох 500

Основний висновок, який можна зробити за допомогою даного моделювання – використання пам’яті у даному типі нейронних мереж потребує більшого часу на навчання нейронної мережі, тобто більше ресурсів.

3.4 Висновки до розділу

Було проведено оцінку двох тестових наборів даних за допомогою нейронних мереж з довгою короткостроковою пам’яттю. Результати моделювання демонструють, що похибки прогнозу для тестових даних зменшуються відповідно до кількості епох, тобто відповідно до часу, наданого мережі на навчання. Таким чином для розглянутих датасетів значення метрик

RMSE, MAPE, DW та MAE покращувались при збільшенні кількості епох, але виявилось, що даній моделі нейронної мережі недостатньо часу для побудови залежностей з пам’яттю при кількості епох 1000. Тим не менш, моделювання за допомогою нейронної мережі дозволяє доволі точно відтворити поведінку часового ряду та визначити збільшення чи зменшення відповідних показників, що дуже важливо при прогнозуванні, як фінансовому, так і метеорологічному.

4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

4.1 Постановка задачі

У даному розділі проводиться функціонально – вартісний аналіз, що є одним із методів евристичного аналізу, який має мету вибору оптимального варіанта, який буде забезпечувати повноцінне виконання досліджуваним об'єктом (у даному випадку, у ролі досліджуваного об’єкта виступає програмний продукт) своїх основних функцій при мінімальних затратах програмного продукту, розробленого для прогнозування часових рядів за допомогою рекурентних нейронних мереж LSTM. Програмний продукт створено на мові програмування Python, з використанням середовища розробки PyCharmCommunity.

4.2 Обґрунтування функцій та параметрів програмного продукту

Оскільки цільовий програмний продукт призначений для прогнозування часових рядів у різноманітних типах задач, основні функції програмного продукту можна виділити наступним чином:

F1 – вибір початкових даних

F2 – обробка тестових даних

F3 – демонстрація результатів роботи програмного продукту

Функція F1:

а) самостійна генерація даних за математичними законами

б) використання реальних вибірок даних

Функція F2:

а) класичні математичні моделі

б) нейронні мережі

Функція F3:

а) використання середовища розробки або консолі для демонстрації результатів

б) розробка програмного інтерфейсу

У морфологічній карті (Рисунок 4.1) наведено варіанти реалізації основних функцій

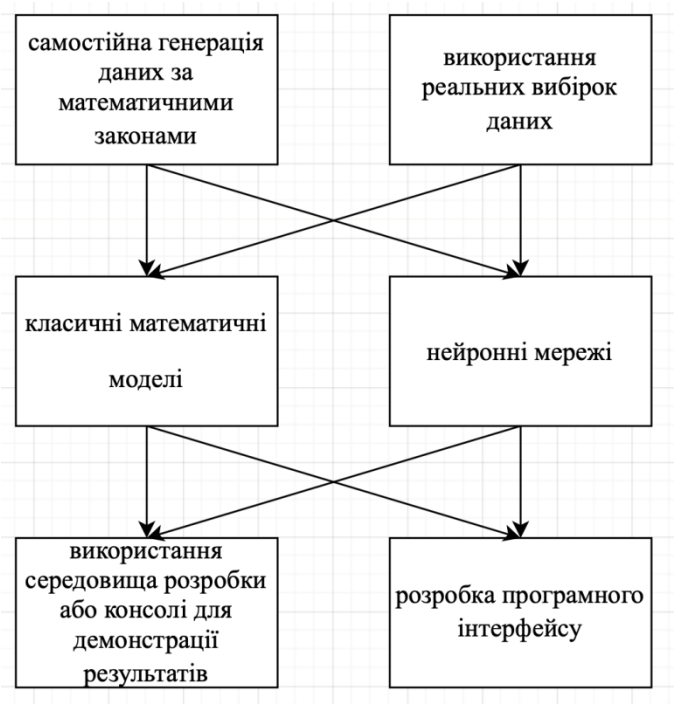


Рисунок 4.1 Морфологічна карта

Згідно з картою було побудовано позитивно-негативну матрицю (Таблиця 4.1)

Таблиця 4.1 Позитивно-негативна матриця

Основна функція	Варіант реалізації	Переваги	Недоліки
F1	А	Відносна швидкість появи даних та незалежність від реальних умов	Цінність даних надто суперечлива для подальшого використання для моделювання реальних процесів
	Б	Точність моделей при застосуванні на реальних даних	Складність отримання такого типу інформації
F2	А	Швидкість роботи	Складність побудови математичних моделей
	Б	Мала похибка отриманих результатів	Час роботи програми
F3	А	Швидкість роботи	Неможливість вручання в процесі моделювання
	Б	Можливість динамічного втручання у роботу програми	Складність реалізації

За результатами аналізу залишаємо наступні варіанти:

F1А –F2Б – F3А

F1Б – F2Б – F3А

Для оцінки прототипу програмного додатку використовуємо параметри, що будуть описані нижче.

4.3 Основні параметри ПП

Оскільки параметрів більше ніж функцій, опишемо взаємозв’язок між функціями та параметрами:

Функція F1 (вибір початкових даних) залежить від кількості записів у тестових даних да кількості вибірок, які надаються для прогнозування. Ці параметри представимо як X1 та X2 відповідно (таблиця 4.2).

Функція F2 (обробка тестових даних) залежить також від кількості вибірок для прогнозування (описаний параметр X2), а також від швидкості виконання моделювання програмою та часу, необхідного для реалізації моделі. Ці параметри представимо як X3 та X4 відповідно (таблиця 4.2).

Функція F3 (демонстрація результатів роботи програмного продукту) залежить від швидкості виконання моделювання програмою та часу, необхідного для реалізації моделі, тобто від описаних раніше параметрів X3 та X4.

Таблиця 4.2 Система програмного продукту

Назва параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Кількість записів у тестових даних	X1	шт	300	100	50
Кількість вибірок для прогнозування	X2	шт	5	10	15
Швидкість виконання моделювання програмою	X3	с	30	15	5
Час, необхідний для реалізації моделі	X4	год	20	10	5

Використовуючи дані таблиці, будуються графічні представлення параметрів (Рисунок 4.2 – 4.5)

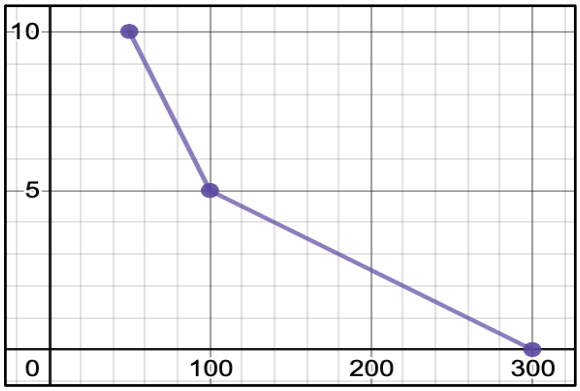


Рисунок 4.2 – Значення параметра X1

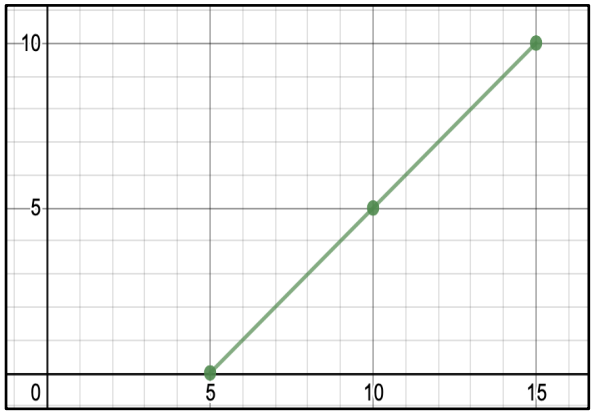


Рисунок 4.3 – Значення параметра X2

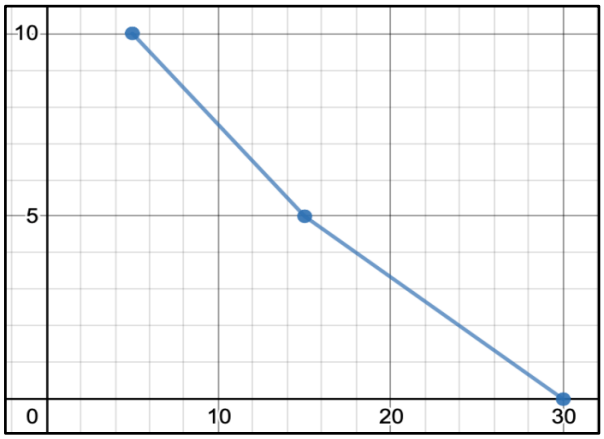


Рисунок 4.4 – Значення параметра X3

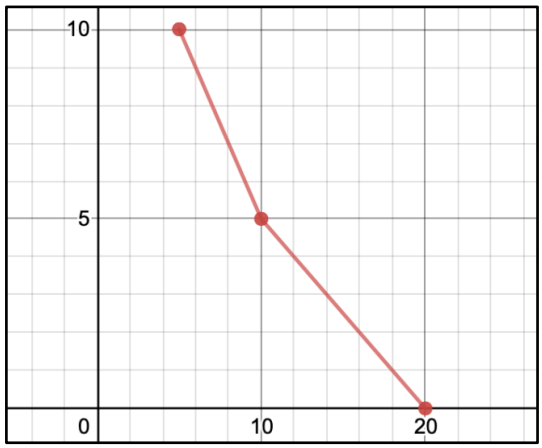


Рисунок 4.5 – Значення параметра X4

Вагомість параметрів оцінюється за допомогою методів попарного зрівняння. Ранги варіюються від 1 до 4. Результати наведені в таблицях 4.3 – 4.4.

Таблиця 4.3 Результат оцінки параметрів

Назва параметра	Умовні позначення	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	$\Delta_i{}^2$
			1	2	3	4	5	6	7			
Кількість записів у тестових даних	X1	шт	1	1	1	1	2	1	2	9	-8.5	72.25
Кількість вибірок для прогнозування	X2	шт	2	2	2	2	1	3	1	12	-5.5	30.25
Швидкість виконання моделювання програмою	X3	с	4	4	4	4	4	4	4	28	10.5	110.25
Час, необхідний для реалізації моделі	X4	год	3	3	3	3	3	2	3	20	2.5	6.25
	Разом		10	10	10	10	10	10	10	70	0	219

Коефіцієнт узгодженості дорівнює

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 219}{49 \cdot (64 - 4)} = 0{,}894 > 0{,}67 \qquad (4.1)$$

Ранг 1 приймається за найменший, а ранг 4 – за найвищий, тобто експерт для найменш важливого на його думку параметра поставить ранг 1, відповідно найважливішому – ранг 4.

Таблиця 4.4 Попарне зрівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	<	<	<	>	<	>	<	0.5
X1 і X3	<	<	<	<	<	<	<	<	0.5
X1 і X4	<	<	<	<	<	<	<	<	0.5
X2 і X3	<	<	<	<	<	<	<	<	0.5
X2 і X4	<	<	<	<	<	>	<	<	0.5
X3 і X4	>	>	>	>	>	>	>	>	1.5

Використовуючи результати попарного порівняння обчислюється вагомість кожного з критеріїв (Таблиця 4.5)

Таблиця 4.5 Розрахунок вагомості параметрів

Xi	ПараметриXj,				Перша ітерація		Друга ітерація		Третя ітерація	
	X1	X2	X3	X4	$в_i$	$K_{вi}$	$в'_i$	$K'_{вi}$	$в''_i$	$K''_{вi}$
X1	1.0	0.5	0.5	0.5	2.5	0.156	9.25	0.157	34.125	0.158
X2	1.5	1.0	0.5	0.5	3.5	0.219	12.25	0.208	44.875	0.207
X3	1.5	1.5	1.0	1.5	5.5	0.344	21.25	0.360	77.875	0.361
X4	1.5	1.5	0.5	1.0	4.5	0.281	16.25	0.275	59.125	0.273
Всього					16	1	59	1	216	1

Проведемо розрахунок рівня якості варіантів реалізації (Таблиця 4.6)

Таблиця 4.6 Розрахунок рівня якості варіантів реалізації

Основна функція	Варіант реалізації	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт якості
F1	А	X1	110	4.9	0.158	0.774
		X2	6	1	0.207	0.207
F1	Б	X1	150	3.9	0.158	0.616
		X2	8	3	0.207	0.621
F2	Б	X2	7	2	0.207	0.414
		X3	20	3.2	0.361	1.155
		X4	6	9	0.273	2.457
F3	А	X3	15	5	0.361	1.805
		X4	7	8	0.273	2.184

$$K1 = 0.774 + 0.207 + 0.414 + 1.155 + 2.457 + 1.805 + 2.184 = 8.996$$

$$K2 = 0.616 + 0.621 + 0.414 + 1.155 + 2.457 + 1.805 + 2.184 = 9.252$$

Оскільки варіант 2 має більший коефіцієнт якості, він є кращим.

4.4 Економічний аналіз варіантів розробки ПП

Варіанти реалізації продукту можна описати наступним чином:

а) Генерування тренувальних даних

б) Пошук реальних даних

Побудова та навчання нейронної мережі

Демонстрація результатів

Для завдання 1 маємо (при реалізації першого варіанту): алгоритм групи складності 2, ступінь новизни В, вид використаної інформації БД

$$Т_р = 19, К_п = 0.6, К_{ск} = 1.07, К_{ст.м} = 1.3$$

$$Т_{1а} = 19 \cdot 0.6 \cdot 1.07 \cdot 1.3 = 15{,}857 \text{ людино-днів.}$$

Для завдання 1 маємо (при реалізації другого варіанту): алгоритм групи складності 3, ступінь новизни Г, вид використаної інформації БД

$$Т_р = 8, К_п = 0.3, К_{ск} = 1.16, К_{ст.м} = 1.4$$

$$Т_{1б} = 8 \cdot 0.3 \cdot 1.16 \cdot 1.4 = 3.898 \text{ людино-днів.}$$

Для завдання 2 маємо алгоритм групи складності 1, ступінь новизни Б, вид використаної інформації ПІ

$$Т_р = 64, К_п = 2.02, К_{ск} = 1.08, К_{ст.м} = 1.45$$

$$Т_2 = 64 \cdot 2.02 \cdot 1.08 \cdot 1.45 = 202.452 \text{ людино-днів.}$$

Для завдання 3 маємо алгоритм групи складності 2, ступінь новизни Г, вид використаної інформації БД

$$Т_р = 12, К_п = 0.36, К_{ск} = 1.0, К_{ст.м} = 1.2$$

$$Т_3 = 12 \cdot 0.36 \cdot 1.0 \cdot 1.2 = 5.184 \text{ людино-днів.}$$

Тоді маємо для варіанту А:

$$\text{Т} = (15{,}857 + 202.452 + 5.184) \cdot 8 = 1787.944 \text{ людино-годин}$$

Тоді для варіанту Б:

$$\text{Т} = (3.898 + 202.452 + 5.184) \cdot 8 = 1692.272 \text{ людино-годин}$$

В розробці бере участь один програміст з окладом 10500 грн та аналітик з окладом 11000 грн

Розрахунок середньої заробітної плати за годину:

$$\text{С} = \frac{21500}{2 \cdot 21 \cdot 8} = 63.988 \text{ грн} \quad (4.2)$$

Заробітна плата для кожного з варіантів реалізації:

$$\text{С1} = 63.988 \cdot 1787.944 = 114407.131 \quad (4.3)$$

$$\text{С2} = 63.988 \cdot 1692.272 = 108285.101 \quad (4.4)$$

Відрахування ЄСВ складають 22%, або для кожного варіанту:

$$\text{С}_{\text{від1}} = 0{,}22 \cdot 114407.131 \text{ грн} = 25169.5688 \text{ грн}$$

$$\text{С}_{\text{від2}} = 0{,}22 \cdot 108285.101 \text{ грн} = 23822.7222 \text{ грн}$$

Визначимо витрати котра потрібна на оплату однієї машино-години. Враховуючи, що заробітна плата одного програміста складає 10 500 грн з коефіцієнтом зайнятості 0,25

$$С_г = 12 \cdot М \cdot К_з = 12 \cdot 10500 \cdot 0.25 = 37500 \text{ грн}$$

Враховуючи додаткову заробітну плату:

$$С_{зп} = С_г \cdot (1 + К_з) = 37500 \cdot 1.25 = 46875 \text{ грн} \quad (4.5)$$

Відрахування на соціальний внесок складатимуть:

$$С_{від} = 0{,}22 \cdot 46875 = 10312.5 \text{ грн} \quad (4.6)$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 11000 грн.

$$С_А = К_{тм} \cdot К_а \cdot Ц_{пр} = 1.15 \cdot 0.25 \cdot 11000 = 3162.5 \text{ грн}$$

Розраховуємо витрати на профілактику та ремонт:

$$С_р = К_{тм} \cdot К_р \cdot Ц_{пр} = 1.15 \cdot 0.05 \cdot 11000 = 632.5 \text{ грн}$$

Розраховуємо ефективний годинний фонд часу ПК за рік:

$$Т_{ЕФ} = (365 - 104 - 11 - 6) \cdot 8 \cdot 0.9 = 1756.8 \text{год}$$

Розраховуємо витрати на електроенергію:

$$С_{ЕЛ} = Т_{ЕФ} \cdot N_{С} \cdot К_{З} \cdot Ц_{ЕН} = 1756.8 \cdot 0.25 \cdot 0.954 \cdot 1.75 = 733.244\text{грн} \quad (4.7)$$

Розраховуємо накладні витрати:

$$С_{Н} = 11000 \cdot 0.67 = 7370\text{грн} \quad (4.8)$$

Розраховуємо річні експлуатаційні витрати:

$$С_{ЕКС} = 46875 + 10312.5 + 3162.5 + 632.5 + 733.244 = 61715.744\text{грн} \quad (4.9)$$

Розраховуємо собівартість однієї машино-години

$$С_{М-Г} = \frac{61715.744}{1756.8} = 35.130\text{грн/год} \quad (4.10)$$

Розраховуємо оплату машинного часу в залежності від варіантів:

$$С_{М1} = 35.130 \cdot 1787.944 = 62810.472 \text{ грн}$$
$$С_{М2} = 35.130 \cdot 1692.272 = 59449.515 \text{ грн}$$

Розраховуємо накладні витрати в залежності від варіантів:

$$С_{Н1} = 114407.131 \cdot 0.67 = 76652.778 \text{ грн} \quad (4.11)$$

$$С_{Н2} = 108285.101 \cdot 0.67 = 72550.95 \text{ грн} \quad (4.12)$$

Розраховуємо вартість розробки в залежності від варіантів:

$$С_{ПП1} = 114407.131 + 25169.5688 + 62810.472 + 76652.778 = 279039.950 \text{ грн} \quad (4.13)$$

$$С_{ПП2} = 108285.101 + 23822.7222 + 59449.515 + 72550.95 = 264108.288 \text{ грн} \quad (4.14)$$

Розраховуємо коефіцієнти техніко-економічного рівня для кожного варіанта за формулою:

$$К_{ТЕРj} = \frac{К_{Кj}}{С_{Фj}} \quad (4.15)$$

Для першого варіанта

$$К_{ТЕР1} = \frac{8.996}{279039.950} = 3.22 \cdot 10^{-5} \quad (4.16)$$

Для другого варіанта

$$К_{ТЕР2} = \frac{9.252}{264108.288} = 3.503 \cdot 10^{-5} \quad (4.17)$$

4.5 Висновки до розділу

Отже, можемо зробити висновок, що найбільш ефективним буде варіант з коефіцієнтом техніко-економічного рівня $3.503 \cdot 10^{-5}$, тобто другий варіант. Таким чином, після проведеного функціонально – вартісного аналізу було прийняте рішення реалізовувати варіант з реальними даними для навчання, використовувати для моделювання нейронні мережі та результати моделювання подавати у виді записів у консолі.

ВИСНОВКИ ПО РОБОТІ

Дана робота була присвячена аналізу і дослідженню роботи нейронних мереж з довгою короткостроковою пам’яттю на прикладі задачі прогнозування часових рядів.

В роботі була побудована модель нейронної мережі з довгою короткостроковою пам’яттю для прогнозування часових рядів, виконана оцінка роботи моделі за допомогою статистичних показників.

Результати аналізу та порівняння побудованої моделі вказують на те, що нейронні мережі з довгою короткостроковою пам’яттю мають змогу прогнозувати значення і поведінку досліджуваних часових рядів при належному часі на навчання та достатньому об’єму даних.

Також необхідно зазначити, що час, необхідний мережам на навчання, є досить великим, що означає, що для більш точного прогнозування необхідно виділити більше часу на навчання моделі або використовувати біль потужні обчислювальні машини, які мали б змогу за той самий час навчити модель швидше.

ПЕРЕЛІК ПОСИЛАНЬ

1. Берзлев О.Ю. Сучасний стан інформаційних систем прогнозування часових рядів. ДВНЗ «Ужгородський національний університет», Ужгород, 2013. С. 78. URL: http://urss.knuba.edu.ua/files/zbirnyk-13/78-82.pdf (дата звернення: 11.04.2020).
2. Чернодуб А.М. Навчання рекурентних нейронних мереж методом псевдорегуляризації для багатокрокового прогнозування часових рядів, Математичні машини і системи, 2012, No 4. С. 41. URL: https://cyberleninka.ru/article/n/navchannya-rekurentnih-neyronnih-merezh-metodom-psevdoregulyarizatsiyi-dlya-bagatokrokovogo-prognozuvannya-chasovih-ryadiv/viewer (дата звернення: 11.04.2020).
3. Jason Brownlee, Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras. URL: https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/ (Last accessed: 12.04.2020).
4. Прогнозування та аналіз часових рядів. Методичні вказівки до практичних занять та самостійної роботи студентів спеціальності 051 «Економіка» освітня програма «Економічна кібернетика», «Економічна аналітика» / Уклад.: Юрченко М. Є. Чернігів: ЧНТУ, 2018. 88 с.
5. Кучанський О. Ю., Білощицький А. О. Прогнозування часових рядів методом селективного зіставлення зі зразком: *Восточно-Европейский журнал передовых технологий*, 2015. С. 13.
6. Сергеева Л.Н. Моделирование поведения экономических систем методами нелинейной динамики (теории хаоса). Запорожье: ЗГУ, 2002, 227 с.
7. Головко В. А. Нейронные сети: обучение, организация и применение. Кн.4: Учеб.пособие для вузов / за ред. А.И. Галушкина. / В. А. Головко. Москва: ИПРЖР, 2001, 256 с.

8. Розенблат Ф. Принципы нейродинамики: Персептрон и теория механизмов мозга. Пер. з англ. / Ф. Розенблат. Москва: Мир, 1965. 470 с.
9. Белокопытов Ю.Н. Человеческий мозг как фрактальная голограмма внешнего нелинейного мира: ВЕСТНИК ИрГТУ No8 (67) 2012. С. 232 – 233.
10. Горбачевская Е.Н. Классификация нейронных сетей. URL: https://cyberleninka.ru/article/n/klassifikatsiya-neyronnyh-setey/viewer) (дата звернення: 19.04.2020.
11. Чабаненко Д. М. Математичні моделі та методи прогнозування часових рядів на основі складних ланцюгів Маркова: дис. ... канд. техн. наук: 01.05.02/Нац. ун-т ім. Богдана Хмельницького. Черкаси, 2012. 183 с.
12. Качаловський А.С. Методи і моделі прогнозування лінійних та нелінійних нестаціонарних процесів: *"Інтернаука"*. URL: https://www.inter-nauka.com/uploads/public/14655658092305.PDF (дата звернення: 20.04.2020).
13. Visscher P.E. Forecasting Time Series with Artificial Neural Networks: Master Thesis / Mathematical Institute Utrecht University. Utrecht, 2018, 87 p.
14. Christopher Olah: Understanding LSTM Networks. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (дата звернення: 15.04.2020).
15. ДСТУ 3008:2015. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення. [На заміну ДСТУ 3008-95; чинний від 2015-06-22]. Вид. офіц. Київ: ДП «УкрНДНЦ», 2016. 31 с. (Інформація та документація).

ДОДАТОК А

Лістинг програми

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math
import keras as kr
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.tsa.arima_model import ARIMA as ARIMA_old
from statsmodels.stats.stattools import durbin_watson as DW

#We should specify the name of csv file
filename = 'comp_obf.csv'

def create_model(state, stepvalue):
    model = kr.models.Sequential()
    model.add(kr.layers.LSTM(4,
              input_shape=(1, stepvalue),
              activation='relu',
              batch_size=1,
              stateful=state))
    model.add(kr.layers.Dense(1, activation='relu'))
    model.compile(loss='mse', optimizer='adam')
    return model
```

```
def show_func(filename):
    np.random.seed(1)
    Matrix = pd.read_csv(filename, sep=';', usecols=[0, 1], engine='python')
    print(Matrix)
    model = ARIMA(Matrix.values, order=(2, 2, 2))
    model_old = ARIMA_old(Matrix.values, order=(2, 2, 2))
    model_fit = model.fit()
    model_old_fit = model_old.fit(disp=0)
    print(model_fit.summary())
    print('Model RMSE:')
    print(math.sqrt(model_fit.mse))
    print('Model DW:')
    print(DW(model_fit.resid))
    print('Model MAE:')
    print(model_fit.mae)
    residuals = pd.DataFrame(model_fit.resid)
    fig, ax = plt.subplots(1, 2)
    residuals.plot(title="Residuals", ax=ax[0])
    residuals.plot(kind='kde', title='Density', ax=ax[1])
    plt.show()
    #model_old_fit.plot_predict(dynamic=False)
    plt.show()
    return

def prediction_func(filename, stepvalue, state):
    np.random.seed(1)
    # load the dataset
    Matrix = pd.read_csv(filename, sep=';', usecols=[0, 1], engine='python')
```

```
print(Matrix)
#this part is needed for xlsx file reading
"""
xls_file = pd.ExcelFile('data.xlsx')
print(xls_file.sheet_names)
df = xls_file.parse(xls_file.sheet_names[0])
print(df)
for line in range(len(df['slr'])):
    if pd.isna(df['slr'][line]):
        df['slr'][line] = 0
#print(df)
matrix_not_cutted = df[1400:14730]
matrix_not_cutted = matrix_not_cutted.head(100)
print(matrix_not_cutted)
Series = matrix_not_cutted['slr'].values
Series = Series.reshape(-1, 1)
"""
# converting into array
Series = Matrix.values
Series = Series.astype('float32')
scaler = MinMaxScaler(feature_range=(0, 1))
Series = scaler.fit_transform(Series)
# dividing into train and test
lentrain = int(len(Series) * 0.73)
train = Series[0:lentrain, :]
test = Series[lentrain:len(Series), :]
X = []
Y = []
range_train = len(train) - stepvalue - 1
```

```
for i in range(range_train):
    Y.append(train[i + stepvalue, 0])
    X.append(train[i:(i + stepvalue), 0])
trX = np.array(X)
trY = np.array(Y)
range_test = len(test) - stepvalue - 1
X = []
Y = []
for i in range(range_test):
    Y.append(test[i + stepvalue, 0])
    X.append(test[i:(i + stepvalue), 0])
tX = np.array(X)
tY = np.array(Y)
trX = np.reshape(trX, (trX.shape[0], 1, trX.shape[1]))
tX = np.reshape(tX, (tX.shape[0], 1, tX.shape[1]))
model = create_model(state, stepvalue)
if state:
    for i in range(500):
        print('real epoch: '+str(i))
        model.fit(trX, trY, epochs=1, batch_size=1, verbose=2, shuffle=False)
        model.reset_states()
    tr_pred = model.predict(trX, batch_size=1)
    model.reset_states()
    test_pred = model.predict(tX, batch_size=1)
else:
    model.fit(trX, trY, epochs=100, batch_size=1, verbose=2)
    tr_pred = model.predict(trX)
    test_pred = model.predict(tX)
tr_pred = scaler.inverse_transform(tr_pred)
```

```
trY = scaler.inverse_transform([trY])
test_pred = scaler.inverse_transform(test_pred)
tY = scaler.inverse_transform([tY])
MAPE_sum1 = 0
MAPE_sum2 = 0
err_train = []
err_test = []
for i in range(len(trY[0])):
    MAPE_sum1 += (abs((trY[0][i] - tr_pred[:, 0][i]))/trY[0][i])
    err_train.append(trY[0][i] - tr_pred[:, 0][i])
MAPE_train = MAPE_sum1/len(trY[0])
for i in range(len(tY[0])):
    MAPE_sum2 += (abs((tY[0][i] - test_pred[:, 0][i]))/tY[0][i])
    err_test.append(tY[0][i] - test_pred[:, 0][i])
MAPE_test = MAPE_sum2/len(tY[0])
RMSE_train = math.sqrt(mean_squared_error(trY[0], tr_pred[:, 0]))
print('Train Score RMSE:')
print(RMSE_train)
print('Train Score DW:')
print(DW(err_train))
MAE_train = mean_absolute_error(trY[0], tr_pred[:, 0])
print('Train Score MAE:')
print(MAE_train)
print('Train Score MAPE:')
print(MAPE_train)
RMSE_test = math.sqrt(mean_squared_error(tY[0], test_pred[:, 0]))
print('Test Score RMSE:')
print(RMSE_test)
print('Test Score DW:')
```

```
    print(DW(err_test))
    MAE_test = mean_absolute_error(tY[0], test_pred[:, 0])
    print('Test Score MAE:')
    print(MAE_test)
    print('Test Score MAPE:')
    print(MAPE_test)
    res = pd.DataFrame(err_train + err_test)
    real_plot = pd.DataFrame(scaler.inverse_transform(Series))
    real_plot.plot(title="Real Data")
    #res.plot(title="Residuals")
    fig, ax = plt.subplots(1, 2)
    res.plot(title="Residuals", ax=ax[0])
    res.plot(kind='kde', title='Density', ax=ax[1])
    plt.show()
    Plot1 = np.empty_like(Series)
    Plot1[:, :] = np.nan
    Plot1[stepvalue:len(tr_pred) + stepvalue, :] = tr_pred
    Plot2 = np.empty_like(Series)
    Plot2[:, :] = np.nan
    Plot2[len(tr_pred) + (stepvalue * 2) + 1:len(Series) - 1, :] = test_pred
    plt.plot(scaler.inverse_transform(Series))
    plt.plot(Plot1, color='red')
    plt.plot(Plot2, color='magenta')
    plt.show()
    return

show_func(filename)
prediction_func(filename, 1, True)
```

ДОДАТОК Б

Ілюстративний матеріал

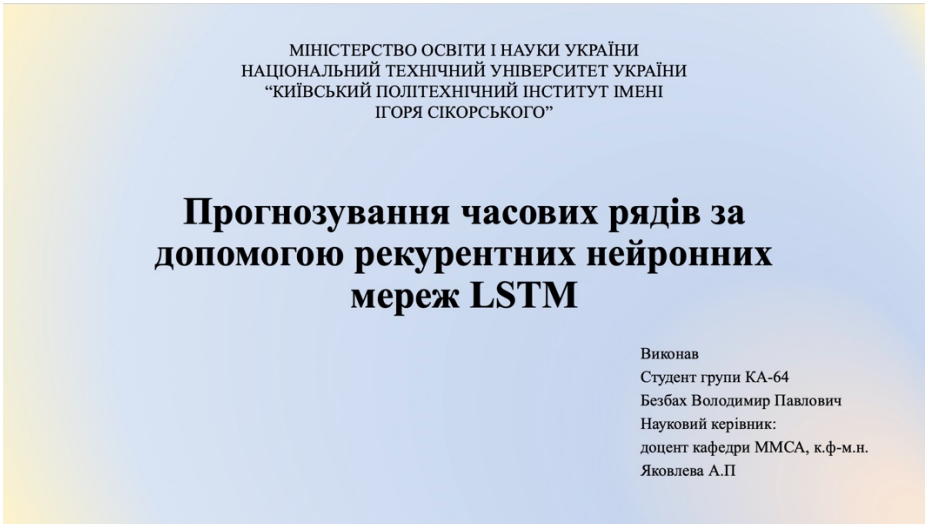


Рисунок 1 – Вступний слайд

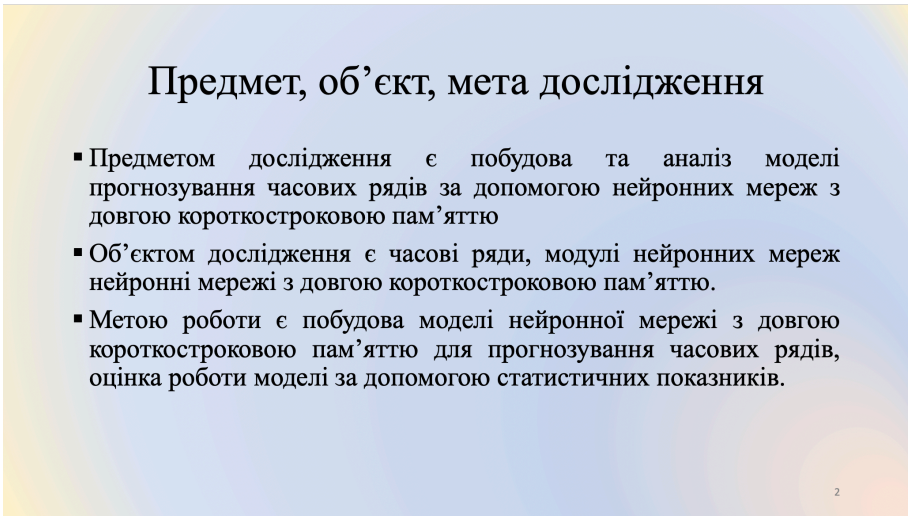


Рисунок 2 – Опис предмету, об’єкта та мети дослідження

Актуальність теми

Прогнозування часових рядів є актуальною науковою проблемою, що має безліч застосувань у теорії управління, в економіці, медицині, фізиці та інших галузях. Якщо казати про нейромережеві методи, то вони добре себе зарекомендували як засіб моделювання динамічних систем при невідомій апріорі математичній моделі динамічної системи.

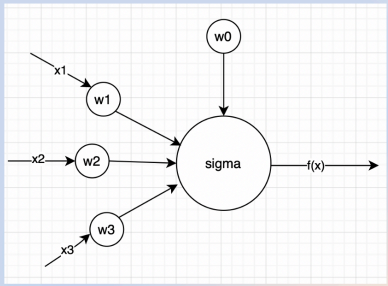
Сьогодні інформаційні системи прогнозування є невід'ємними складовими процесів управління складними системами і прийняття управлінських рішень, застосовуються аналітиками для оцінювання ризиків фінансового інвестування тощо

3

Рисунок 3 – Опис актуальності теми

Нейронна мережа

Нейронна мережа – це сукупність нейронних елементів і зв'язків між ними. Основний елемент нейронної мережі – це формальний нейрон, що здійснює операцію нелінійного перетворення суми добутків вхідних сигналів на вагові коефіцієнти. Такий нейрон ще називають одношаровим перцептроном



4

Рисунок 4 – Опис перцептрона у нейронній мережі

Рисунок 5 – Опис багатошарового перцептрона

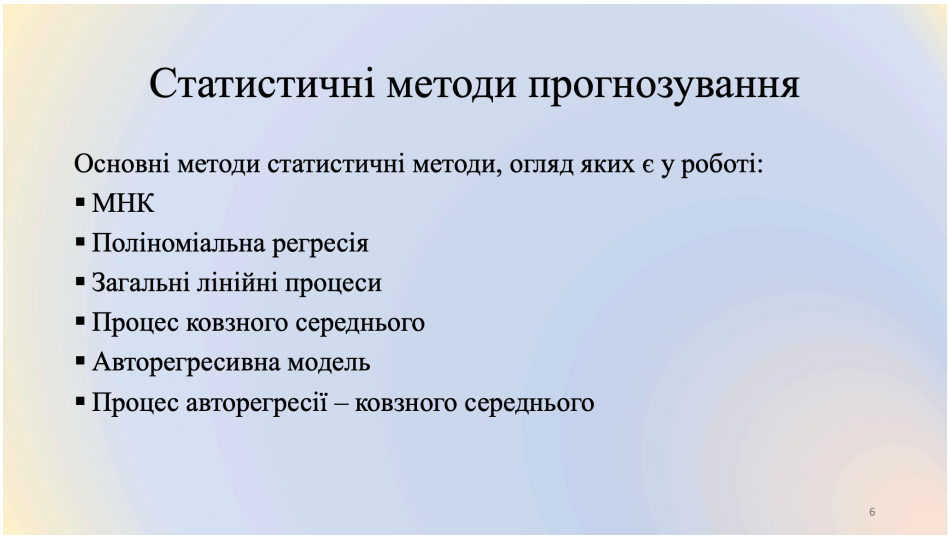


Рисунок 6 – Перелік статистичних методів прогнозування, описаних у роботі

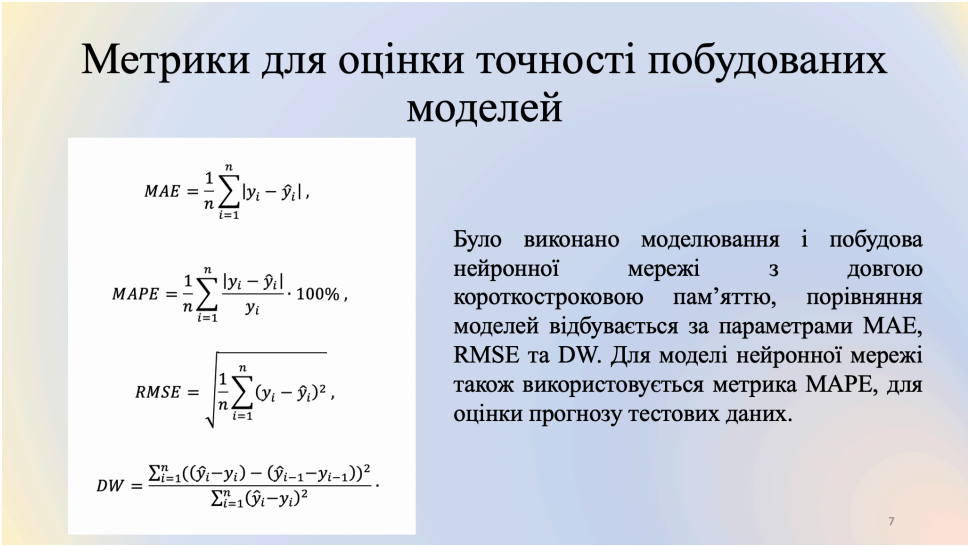


Рисунок 7 – Опис метрик оцінки точності побудованих моделей

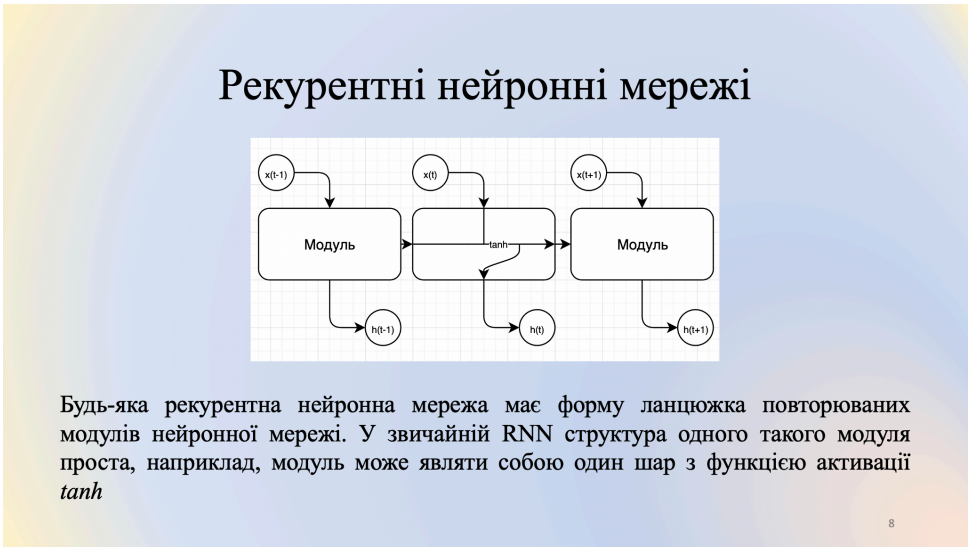


Рисунок 8 – Опис рекурентних нейронних мереж

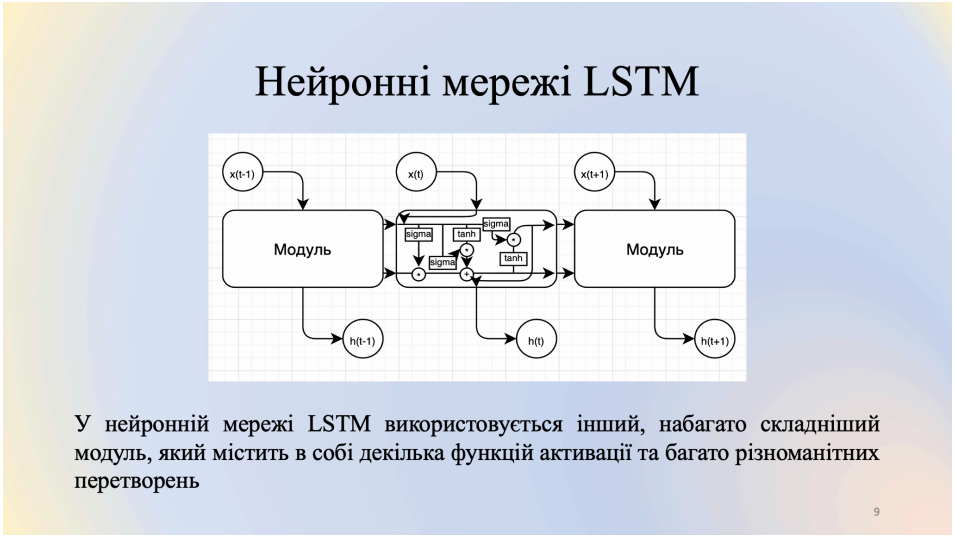


Рисунок 9 – Опис рекурентних нейронних мереж LSTM

Рисунок 10 – Опис стану клітини у нейронній мережі LSTM

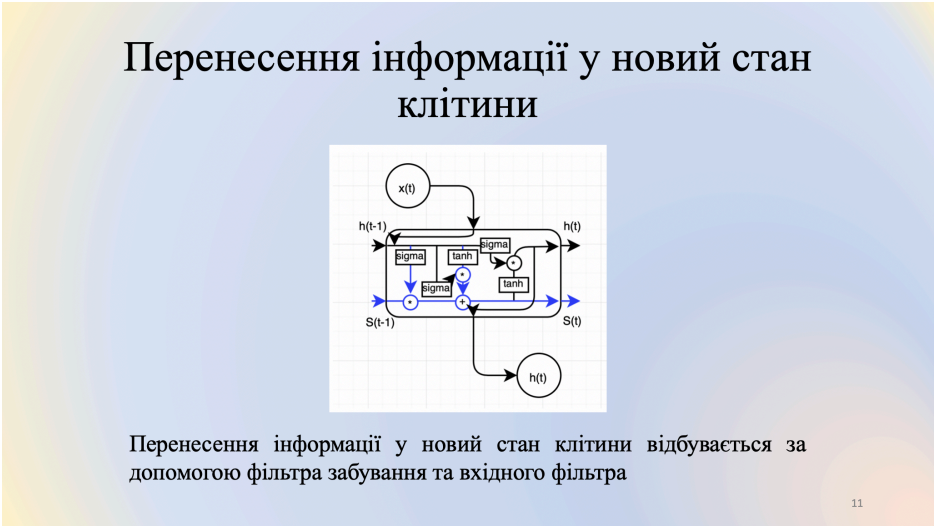


Рисунок 11 – Опис нового стану клітини у нейронній мережі LSTM

Рисунок 12 – Опис нового вихідного сигнала у нейронній мережі LSTM

Рисунок 13 – Опис та приклад обраних даних для моделювання

Рисунок 14 – Результати моделювання на першому наборі даних

Рисунок 15 – Результати моделювання на другому наборі даних

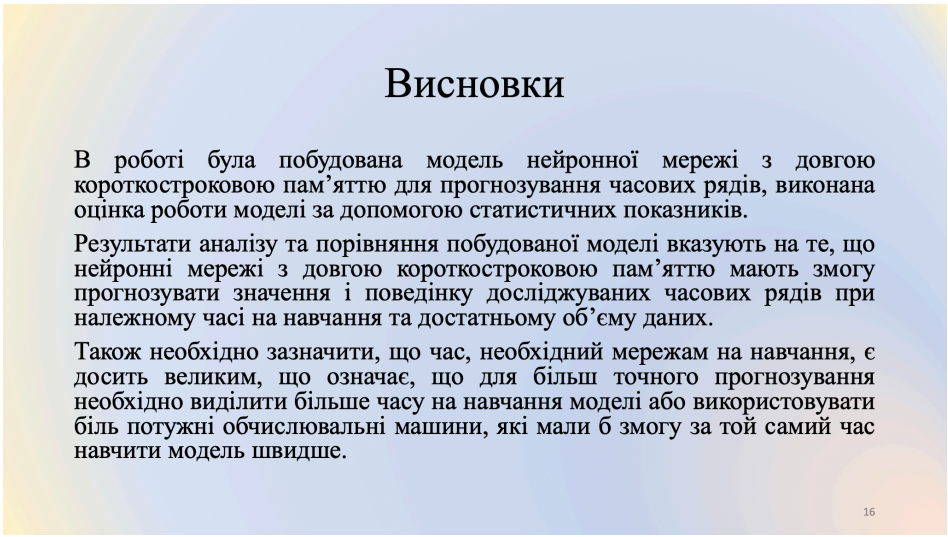


Рисунок 16 – Висновки

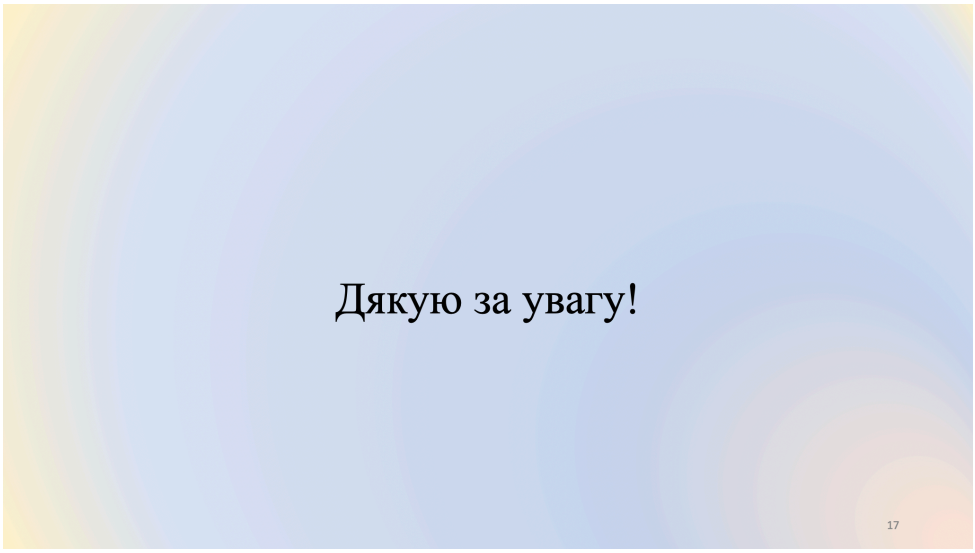


Рисунок 17 – Завершальний слайд